CSC-EPL-89/001

2

NATIONAL COMPUTER SECURITY CENTER

FINAL EVALUATION REPORT OF DATA GENERAL CORPORATION

ADVANCED OPERATING SYSTEM/VIRTUAL STORAGE

(AOS/VS)

22 February 1989

Approved for Public Release:
Distribution Unlimited.

FINAL EVALUATION REPORT

DATA GENERAL CORPORATION

ADVANCED OPERATING SYSTEM/VIRTUAL STORAGE

(AOS/VS)

NATIONAL
COMPUTER SECURITY CENTER

9800 Savage Road
Fort George G. Meade
Maryland 20755-6000

February 22, 1989

CSC-EPL-89/001
Library No. ~~S231,329~~
S236,351

FOREWORD

This publication, the Final Evaluation Report Data General Corporation, Advanced Operating System/Virtual Storage, is being issued by the National Computer Security Center under the authority of and in accordance with DoD Directive 5215.1, "Computer Security Evaluation Center". The purpose of this report is to document the results of the formal evaluation of Data General's Advanced Operating System/Virtual Storage revision 7.60 operating system. The requirements stated in this report are taken from *DEPARTMENT OF DEFENSE TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA* dated December 1985.

Approved:

Eliot Sohmer ______ February 22, 1989
Eliot Sohmer
Chief, Office of Computer Security
Evaluations, Publications, and Support
National Computer Security Center

ACKNOWLEDGEMENTS

Team Members

Team members included the following individuals, who were provided by the following organizations:

Albert C. Hoheb
R. Leonard Brown, Ph.D.
Cynthia Grall

The Aerospace Corporation
El Segundo, California

Santosh Chokhani, Ph.D.

The MITRE Corporation
McLean, Virginia

Joseph Bulger
Donald Dasher

National Computer Security Center
Ft. George G. Meade, Maryland

Further Acknowledgements

Many thanks are extended to the previous developmental team members for their hard work which brought this evaluation into the formal stage.

Table of Contents

EXECUTIVE SUMMARY

The security protection provided by Data General Corporation's Advanced Operating System/Virtual Storage (AOS/VS) revision 7.60 operating system running on the ECLIPSE MV/Family[1] of 32-bit super-minicomputers has been examined by the National Computer Security Center (NCSC). The security features of AOS/VS were examined against the requirements specified by the *DoD Trusted Computer System Evaluation Criteria* (the Criteria), dated December 1985, in order to establish a candidate rating.

The NCSC evaluation team has determined that the highest class at which AOS/VS satisfies all the specified requirements of the Criteria is class C2. AOS/VS, using the specified hardware and software (see Appendices A and B), configured and operated as described in the Trusted Facility Manual, has been assigned a class C2 rating.

A system that has been rated as being a C2 system provides a Trusted Computing Base (TCB) that enforces a discretionary (need-to-know) access control mechanism and audits the security relevant actions of individual users.

[1] ECLIPSE, ECLIPSE MV/4000, ECLIPSE MV/6000, ECLIPSE MV/8000, ECLIPSE MV/15000, ECLIPSE MV/20000, and microNOVA are trademarks of Data General Corporation.

INTRODUCTION

In April 1988, the National Computer Security Center (NCSC) began a formal product evaluation of Data General Corporation's AOS/VS operating system revision 7.60 and ECLIPSE MV/Family of computers. The objective of this evaluation was to rate the AOS/VS computing system against the Criteria and place the product on the Evaluated Products List. It is intended that this report give evidence and analysis of the security features and assurances provided by the AOS/VS computing system. This report documents the evaluation team's understanding of the product's security design and appraises its functionality and integrity against the Criteria's C Division security requirements.

Material for this report was gathered by the NCSC AOS/VS evaluation team through documentation, training, interaction with system developers, and experience gained while using and testing an AOS/VS system.

Evaluation Process Overview

The Department of Defense Computer Security Center was established in January 1981 to encourage the widespread availability of trusted computer systems for use by facilities processing classified or other sensitive information. In August 1985 the name of the organization was changed to the National Computer Security Center. In order to assist in assessing the degree of trust one could place in a given computer system, the DoD Trusted Computer System Evaluation Criteria was written. The Criteria establishes specific requirements that a computer system must meet in order to achieve a predefined level of trustworthiness. The Criteria levels are arranged hierarchically into four major divisions of protection, each with certain security-relevant characteristics. These divisions are, in turn, subdivided into classes. To determine the division and class at which all requirements are met by a system, the system must be evaluated against the Criteria by an NCSC evaluation team.

The NCSC performs evaluations of computer products in varying stages of development from initial design to those that are commercially available. Product evaluations consist of a developmental phase and a formal phase. All evaluations begin with the developmental phase. The primary thrust of the developmental phase is an in-depth examination of a manufacturer's design for either a new trusted product or for security enhancements to an existing product. Since the developmental phase is based on design documentation and information supplied by the industry source, it involves no "hands on" use of the system. The developmental phase results in the production of an Initial Product Assessment Report (IPAR). The IPAR documents the evaluation team's understanding of the system based on the information presented by the vendor. Because the IPAR contains proprietary information, distribution is restricted to the vendor and the NCSC.

Products entering the formal phase must be complete security systems. In addition, the release being evaluated must not undergo any additional development. The formal phase is an analysis of the hardware and software components of a system, all system documentation, and a mapping of the security features and assurances to the Criteria. The analysis performed during the formal phase requires "hands on" testing (i.e., functional testing and, if applicable, penetration testing). The formal phase results in the production of a final report and an Evaluated Products List (EPL) ent y. The final report is a summaiy of the evaluation and includes the EPL rating which indicates the final class at which the product successfully met all Criteria requirements in terms of both features and assurances. The final report and EPL entry are made public.

Document Organization

This report consists of four sections and four appendices. Section 1 is an introduction. Section 2 provides an overview of the system hardware and software architecture. Section 3 provides a mapping between the requirements specified in the Criteria and the AOS/VS features that fulfill those requirements. Section 4 contains the evaluator's comments. The appendices identify specific hardware and software components to which the evaluation applies and provide references to acronyms and manuals used throughout the report.

SYSTEM OVERVIEW

Data General's AOS/VS operating system for the ECLIPSE MV/Family of 32-bit super-minicomputers is a general purpose operating system which provides both batch and online processing in a multi-process, multi-tasking environment. AOS/VS runs on Data General's ECLIPSE MV systems ranging from the MV/4000 to the MV/20000. The MV/20000 supports dual processing.

User identification and authentication is performed through verification of a corresponding username and password. An optional password encryption mechanism is provided. AOS/VS maintains an access control list for each directory and data file. An access control list includes the users who can and cannot access files as well as the access attributes (e.g., read, write, execute). AOS/VS also provides an extensive system audit trail capability. The audit trail captures security relevant events such as file accesses, failed-log attempts, and process creations and terminations to monitor attempted breaches of system security. AOS/VS ensures that when file system objects and memory pages are allocated they do not contain residual data. Object reuse for tapes and non-file system disks has to be performed manually.

The AOS/VS TCB consists of the following components:

- ECLIPSE MV system
- Peripheral Controllers and Devices
- Supervisor and AGENT
- AOS/VS Trusted Processes
- Administrative Interfaces

The only subject supported by AOS/VS is the process. Objects supported by AOS/VS include the following: named objects, which consist of files, directories, logical disk units, devices (i.e., terminals, disks, printers, plotters, tape units and diskettes), pipe files, interprocess communication files, queue files, and AOS/VS processes when acted upon.

AOS/VS History

The development story of the first 32-bit processor at Data General (code named "EAGLE") was documented in the popular book *Soul of a New Machine* by Tracy Kidder. It was the story of competition between two engineering sections of Data General to develop the first 32-bit processor to compete against Digital Equipment Corporation's VAX[1] processors. The "EAGLE" later became the MV/8000 processor.

AOS/VS was initially released in 1981 to run on Data General's new 32-bit processors. Advanced Operating System/Virtual Storage (AOS/VS) evolved from Advanced Operating System (AOS) which was designed to run on Data General 16 bit ECLIPSE processors, particularly, the ECLIPSE C/350 processor.

A desire by Data General to be upward compatible influenced the design of the first and subsequent ECLIPSE MV processors and of AOS/VS. The 32-bit processors have an instruction set which is a superset of the 16-bit processor instruction set, and AOS/VS is a superset of AOS from the perspective of system calls.

The ECLIPSE MV processors also support the 16-bit C/350 ECLIPSE stack (narrow stack) along with a full 32-bit ECLIPSE MV stack which is referred to as the wide stack.

ECLIPSE MV Architecture

The ECLIPSE MV processor line is a series of hardware bases offered by Data General. The computer series under evaluation consists of the following processors: MV/4000, MV/6000, MV/8000, MV/10000, MV/15000, and the dual-processor MV/20000. The basic configuration of these machines contains the following system elements: a central processing unit (CPU), a floating point unit (FPU), a memory control unit (MCU), a memory module, an input/output channel controller (IOC), and a diagnostic remote processor (DRP). These elements are organized around a central system address bus (SA) and system data bus (SD).

[1] VAX is a trademark of Digital Equipment Corporation.

The CPU

The ECLIPSE MV 32-bit CPUs execute instructions, generate logical addresses, translate logical addresses to physical addresses, and perform arithmetic and logical data manipulations. The CPU consists of the arithmetic operations and address generation unit, instruction processor, and address translation unit.

The arithmetic operations and address generation unit contains four gate arrays: ALU (arithmetic logic unit), SHIFT, DEC (decoder), and AG (address generator). The first two provide arithmetical, logical, and rotational functions. The DEC decodes commercial data formats, implements the processor control word, and doe combinatorial logic. The AG pipelines effective address calculations, determining the address for the next instruction while the processor executes the final microcycle of the current cycle. The AG also provides simple addition and subtraction for the unit.

The instruction pipeline and microsequencing unit contains the instruction queue, microsequencer, and microcode control store. The six-word instruction queue pipelines the instruction fetch and an instruction decoder converts each instruction into a starting microcode address and associated control information. In the case of a jump instruction or error, the pipeline is flushed and reloaded. The microsequencer generates the subsequent string of microcode addresses for the microprogram implementing the instruction. The microcode control store, which contains both PROM-based and writable microcode, receives the microcode addresses and produces the appropriate control signals for the rest of the CPU.

The logical-to-physical address translation unit supports the address translation mechanism. This mechanism translates logical addresses used by programs running on the CPU into physical addresses in main memory. A detailed explanation of this process follows the processor overview.

Both the instruction fetch path and the data write/read path in the CPU contain caches that speed up the memory access. The instruction cache brings an instruction stream into the CPU. The physical address provides the cache management function and the data alignment swaps high and low words on the 32-bit data path when required.

The FPU

The ECLIPSE MV floating point processor unit is a separate processor that works with the CPU to process floating-point instructions, intrinsic instructions (arc-cosine, arc-sine, arc-tangent, cosine, sine, tangent, floating number to floating power, square root, exponentiation, natural log, common log, and base 2 log), and 32-bit and 64-bit integer multiply and divide instructions. FPU operation is initiated by the CPU microcode with all data and control information sent by the CPU. Once the FPU has received its starting microcode address (dispatch address) and any required data, it can operate asynchronously and in parallel with the CPU, processing the desired instruction while allowing the CPU to continue with other unrelated operations. It is brought online by the microsequencer subsystem when an appropriate instruction appears in the pipeline.

The MCU

The memory control unit provides error detection, correction, and logging, as well as memory integrity checks. It continually monitors accesses to the memory modules that make up main memory. It appends error-correction bits to memory data as it is written, checks and corrects all single-bit and many double-bit errors as memory data is read, and selectively performs an integrity check on the contents of memory during memory-refresh operations. This latter process is called sniffing and prevents single-bit and many double-bit errors from accumulating and destroying correctable data. If a memory error occurs, the MCU stores both the address and a code indicating the type of error The MCU also monitors parity on the SA and SD buses. If a parity error occurs on one of these, the CPU is informed through an instruction trap.

Memory Modules

The ECLIPSE MV memory system is designed to offer a higher degree of modularity. The memory system supports multiple memory modules of various storage capacities, providing both flexibility and growth. Each memory module consists of one printed circuit board and contains dynamic RAM storage, a control gate array, and assorted support logic. The memory modules provide either four or eight megabytes of actual data storage.

I/O Channel Controller

The I/O channel controller (IOC) is responsible for the control of all I/O devices connected to the processor, including programmed I/O (PIO) and direct memory access (DMA). The CPU sends the IOC commands on the SD and SA buses to set up data transfers to and from main memory. The IOC can perform the data transfer without further intervention from the CPU.

The I/O channels interface with the processor or system elements on the SA and SD buses. The I/O channels interface with the actual I/O devices and their associated controllers on internal buses. The IOC contains gate arrays, static RAMS, and buffers and drivers which interface with the various buses. The hardware controls the translation of addresses and commands and the intermediate storage of data. The IOC can interface directly to either a device (such as an array processor or an analog-to-digital converter) or a device controller (such as a disk drive controller or a LAN controller). In the first case, all actions occur on the board(s) that directly interfaces to the IOC. In the second case, some actions take place on the controller board, and some actions take place on the device it controls. The IOC does not distinguish between these two cases.

The Diagnostic Remote Processor

The diagnostic remote processor (DRP) controls diagnostic and system console functions. There are four types of activities which the DRP performs: PIO operations, system control processor (SCP) functions, system powerup control, and remote functions. The DRP directly controls or interfaces to system devices that can be addressed with PIO commands. The DRP only controls system devices and not all PIO for the entire system. All PIO instructions are sent to the IOC which determines which device to direct the instructions to (DRP or other devices) based on the device codes. Each of the devices appears as an ordinary I/O device with Busy and Done flags and maskable interrupt mechanisms. The DRP also contains much of the SCP and system console which allows an operator to examine and modify CPU and memory status, boot devices, and execute microcode and macrocode programs. The remainder of the SCP resides in the CPU microcode. The DRP is involved in the system powerup sequence as well. It controls the CPU reset line(s) and monitors the CPU RUN line during CPU powerup testing and initialization. Finally, the DRP operating system controls the remote diagnostic functions built into all of the processors. This includes controlling the direction of data flow between the system console, remote diagnostic port, and the user port, allowing control store loads directly from the remote diagnostic port, and changing DRP access passwords. The DRP directly routes the system console interface, user port, and remote diagnostic port. The DRP receives the remote diagnostic commands from these interfaces and performs the necessary actions independent of the rest of the system.

In a C2 installation, the remote port of the DRP should not be attached to any modems or communication lines.

Processor Characteristics

The various processors in the ECLIPSE MV/Family differ from each other in terms of physical memory capacities (up to 64 Mbytes), amount of cache memory (up to 16 Kbytes), and size of instruction cache memory (up to 4 Kbytes). The processor instruction sets are identical with the exception of the MV/20000 processor which, in addition to the standard instruction set, supports dual processing instructions. An ECLIPSE MV processor is able to support up to three IOCs.

The registers supported by the ECLIPSE MV/Family of computers are four general purpose 32-bit registers (AC0, AC1, AC2, AC3), four 64-bit floating point registers (FPAC0, FPAC1, FPAC2, FPAC3), a floating point status register (FPSR), four registers for defining a stack (WSB, WSL, WSP, WFP), a Program Counter (PC), eight Segment Base Registers (SBRs), and a processor status register (PSR).

Addressing modes supported by the ECLIPSE MV system are absolute, program counter relative, and base register relative after any indirection has been resolved.

ECLIPSE MV processors support a wide range of instructions for fixed point computations, floating point computations, stack management, queue management, program flow management, character management, device management, system management, memory management, and instructions for ECLIPSE C/350 compatibility. The processors maintain a 31-bit Program Counter, which includes three bits specifying the current segment. Wraparound occurs within the current segment when incrementing the PC since the segment bits are not changed except by explicit ring-crossing instructions.

Addressing Memory

The ECLIPSE MV/Family of computers provide up to 64 MBytes of physical memory which are used by the Address Translation Unit (ATU) to provide the 4 GByte virtual address space. Memory is word (two bytes) addressable. Pages of logical memory are maintained on disk until the processor needs them in physical memory (a page equals 2 Kbytes). When referring to an instruction or to data that currently resides on disk, the processor moves the page to physical memory. When physical memory is full, the processor may first copy a page from memory to disk before moving the referenced page into memory. To facilitate the operation, the processor maintains tables in memory that determine where a page resides and when to overwrite a page in memory with a page from disk. The processor maintains a table of referenced and modified bits.

To access a memory word or words, the processor translates a logical address to a physical address and accesses the physical page which contains the word or words.

Segment Base Registers

To access a segment, the processor first checks the segment base register (SBR) specified by the logical address. Bit 0 of the SBR controls access to the segment by specifying if the processor can refer to the segment to execute the instruction. If the processor cannot refer to the segment, the processor aborts the instruction and services a segment validity protection fault.

The processor maintains one SBR for each of the eight segments. The SBR (SBR0 to SBR7) contains information which validates segment access and I/O access, specifies a one or two level page table, and specifies for its segment the address of the first entry in the page table.

The SBRs can be modified only with privileged instructions. These instructions must be executed in segment 0, otherwise a protection violation fault occurs.

The SBR format is as follows:

Bit 0 is a validity flag that indicates if this given segment is valid.

Bit 1 is a translation level flag that indicates whether the segment is a one level page table or two level page table.

Bit 2 is an LEF (load effective address) mode flag that indicates if the segment is interpreting the instruction as an LEF instruction or I/O instruction.

Bit 3 is an I/O validity flag that indicates the validity for the segment to execute an I/O instruction.

Bits 4-12 are reserved for internal Data General use.

Bits 13-31 are used to compute the physical address of the page table.

Page Numbers

A page number is the page address shifted right 10 bits. A physical address has the format: bits 0-21 for the page number, bits 22-31 for the word offset.

Page Tables

In each segment, the processor accesses a page table that specifies the status of the pages for the segment in memory. The page table manipulation instructions are Load Page Table Entry (LPTE) and Store Page Table Entry (SPTE). The page table contains a page table entry (PTE) for each page. The PTE, which indicates the validity and type of access for a page, indicates if a page is currently in physical memory and contains information needed to translate a logical address to a physical address.

The page table entry format is:

Bit 0 is a validity flag that indicates if the page reference is valid.

Bit 1 is a memory resident flag that indicates if the accessed page is residing in memory.

Bit 2 is a read access flag that indicates whether the process has read access to the page or not.

Bit 3 is a write access flag that indicates whether the process has write access to a page or not.

Bit 4 is an execute access flag that indicates whether the processor has execute access to a page or not. Bit 4 may be set only if bit 2 or 3 is set.

Bits 5-12 are reserved for future use.

Bits 13-31 identify a page in memory. The physical page address refers to a page containing an instruction and/or data, or refers to a page containing the base of another page table as determined by a one or two level page table translation.

Following a valid segment reference, the processor checks the range of the logical address space within the segment and compares it to the address range of the logical address. Bit 1 of the SBR defines a one or two level page table which specifies the addressing range. When the indirect field equals zero, the absolute or relative address becomes the effective address. The processor translates an effective address to a physical address and accesses the physical address. When the indirect field equals one, the absolute or relative address becomes an indirect address or pointer. The processor translates the indirect address to a physical address and uses the contents of that physical address as another indirect or direct address. The processor continues to resolve pointers until bit 0 is zero. The indirect and effective logical address formats are as follows:

Bit 0 is used to determine indirect addressing. In the base register relative addressing mode, bit 0 is ignored and bits 1 - 31 of the PC, AC2 or AC3 are used in address calculation.

Bits 1-3 are segment bits which specify one of the eight SBRs.

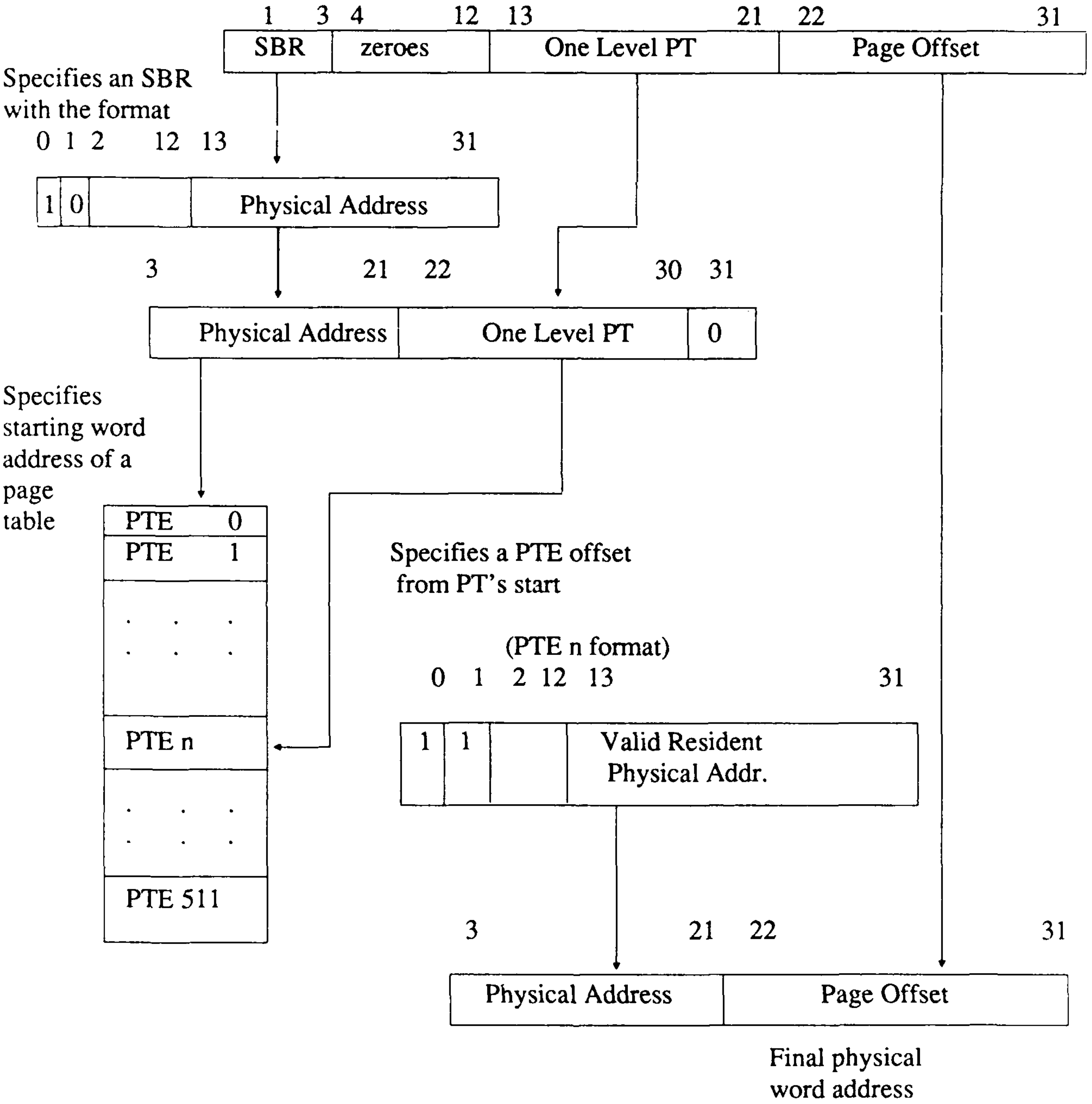
Bits 4-12 specify an offset for the first level of the two level page table entry. For a one level page table translation, these bits must all be zero; otherwise the processor aborts the instruction and services a page table validity protection fault.

Bits 13-21 specify an offset for the final page table entry for both one level and two level page table entries.

Bits 22-31 are page offset bits that specify the word address in the page. The page offset completes the address translation.

One level page table translation (see Figure 1):

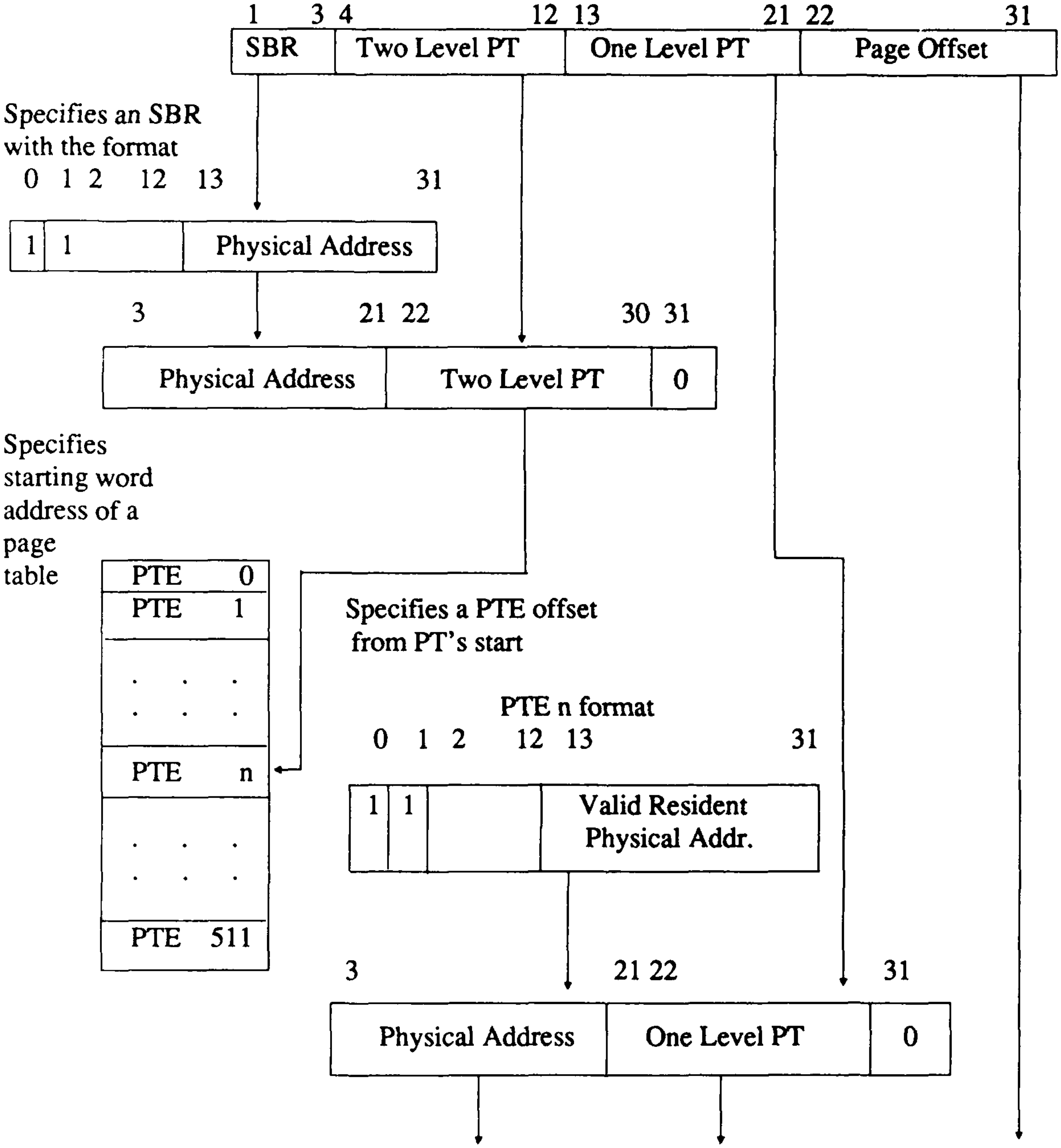
The logical address to be translated has the format previously stated. Bits 1-3 of the address specify one of the eight SBRs. The processor uses the contents of this valid SBR to form the physical address of the PTE. To form this physical page address, the processor begins with the physical address specified in the bits 13-31 of the SBR. This address becomes bits 3-21 of the PTE address. Bits 13-21 of the logical address becomes bits 22-30 of the PTE address. The processor appends a zero to the right of the PTE address, making a 29-bit address. This PTE specifies the starting address of a page of memory. PTEn bits 13-31, the page address, becomes bits 3-21 of the physical address. The page offset field specified in bits 22-31 of the logical address become bits 22-31 of the physical address. This is the physical word address translated from the logical word address.

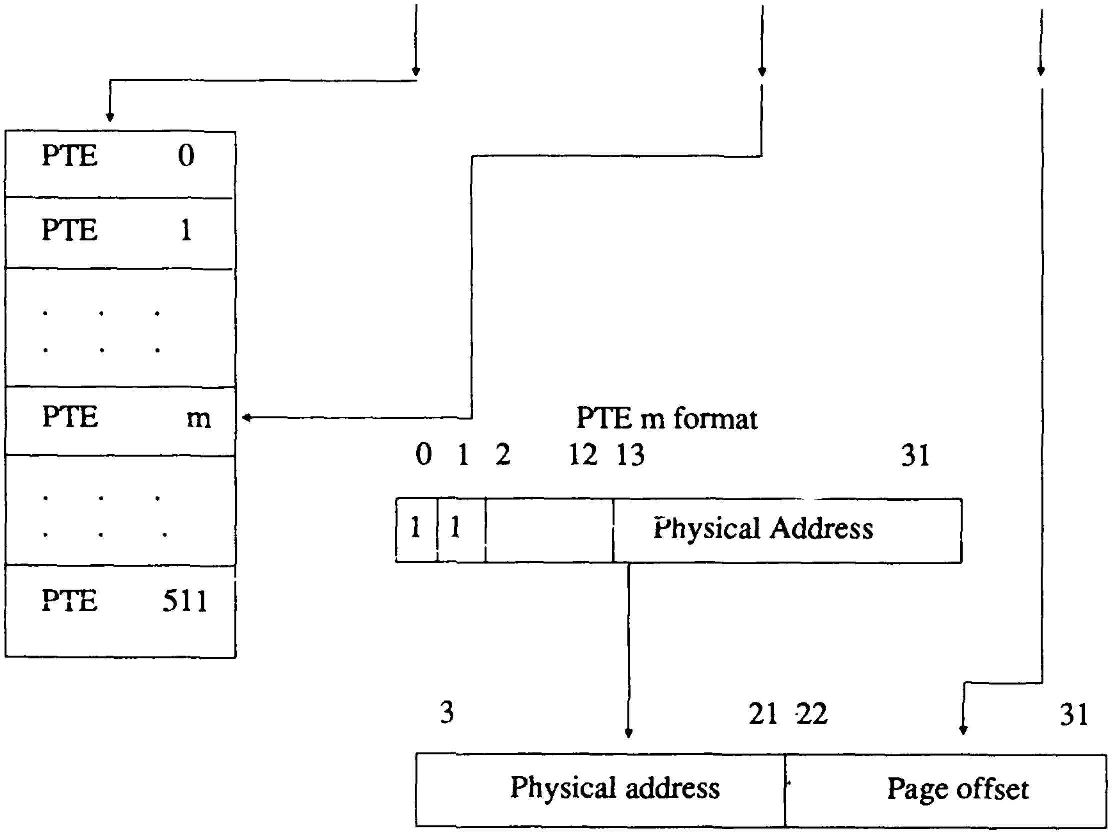


One Level Page Table Translation
Figure 1.

Two level page table translation (see Figure 2):

Bits 1-3 of the logical address specify one of the eight SBRs. The processor uses the contents of the valid SBR to form the address of a PTE. The processor begins with the physical address specified in bits 13-31 of the SBR. This address becomes bits 3-21 of the PTE address. Bits 4-12 of the logical address become bits 22-30 of the PTE address. The processor appends a zero to the right of the PTE address, making a 29-bit address. The PTE specifies the starting address of a page table. The processor now constructs the address of a second PTE. The physical address specified in bits 13-31 of the first (PTEn) become bits 3-21 of the address of the second PTE (labeled PTEm). Bits 13-21 of the logical address become bits 22-30 of the second PTE address. The processor appends a zero to the right of the second PTE address to make a 29-bit address. The second PTE specifies the starting address of a page. The page table containing PTEm can be paged itself. PTEn can indicate a nonresident page table. PTEm's bits 13-31, the page address, become bits 3-21 of the physical address. The page offset specified in bits 22-31 of the logical address become bits 22-31 of the physical address. This last value is the physical word address.


1 3 4 12 13 21 22 31
SBR
Two Level PT
One Level PT
Page Offset
Specifies an SBR
with the format
0 1 2 12 13 31
1
1
Physical Address
3 21 22 30 31
Physical Address
Two Level PT
0
Specifies
starting word
address of a
page
table
PTE 0
PTE 1
PTE n
PTE 511
Specifies a PTE offset
from PT's start
PTE n format
0 1 2 12 13 31
1
1
Valid Resident
Physical Addr.
3 21 22 31
Physical Address
One Level PT
0



Final physical word address

Two Level Page Table Translation
Figure 2.

Memory Reference Instructions

When a memory reference instruction addresses the current segment, the processor compares the page attributes with the type of access the instruction requests, determining the validity of the reference. The page attributes are identified as valid page, read access, write access, and execute access. The processor also compares the segment field of every indirect address reference with the current segment. For accessing data (read or write access), direct and indirect addressing can occur within the current segment or towards a higher numbered segment. For transferring control (execute access), indirect addressing must occur in the current segment; direct addressing can occur within the current segment or other segments using proper ring crossing instructions. Otherwise, the processor aborts the access with a protection violation and services the fault.

The processor will handle up to 15 consecutive indirect address references for any instruction before a protection violation occurs.

Privileged Faults

Upon detection of a privileged fault, the ATU generates either a page fault or protection fault. A page fault occurs when the interpretation of the validity and appropriate access bits in a page table entry is coupled with an attempt to refer to a location that is part of the logical address space but is not part of the physical address space.

Page Faults

When a page fault occurs, the following actions result. The processor is either residing in segment 0 or transfers to segment 0 by storing the current frame pointer and stack pointer in the current segment and crossing into segment 0. The processor then stores the context block using locations 32-33 of segment 0 as the base address. Then the segment 0 stack is initialized and the fault code is stored in AC1. The processor jumps through locations 30-31 of segment 0 to the page fault handler. The page fa'ılt handler selects a page from memory and writes it to disk, loads the referenced page from disk into the memory space vacated, and then restores the state of the processor. The processor completes the memory reference and continues executing the instruction. If a page fault occurs before the processor jumps to the page fault handler, then the processor halts. Page zero of the current segment and page zero of segment zero must be resident. In addition, the context block and page fault handler must always be resident, else the processor halts.

Address Protection Faults

When the address translator is enabled, the system checks for protection violation faults. When a fault occurs, the fault code and associated priority are stored in AC1.

Protection Violations

The protection violations handled by an ECLIPSE MV system include: privileged or I/O instruction violations; indirect addressing violations; inward reference violations; segment validity violations; read, write or execute access violations; page table validity violations; and segment crossing violations.

A protection violation will cause an internal interrupt which will transfer control to ring 0 code to handle the violation appropriately. A user program causing such a protection violation will then be returned an appropriate user trap. User programs may be written such that the traps are handled gracefully; otherwise, the programs will abort. Regardless of how the user program handles the interrupt, the TCB will not allow the protection mechanism to be bypassed.

Ring Architecture

The ECLIPSE MV/Family of computers provide a 4 GByte virtual address space broken up into eight segments. Each segment provides up to a 512 MByte, word-addressed, virtual address space, and is protected by a ring. Rings are (conceptually) bound to corresponding segments. Ring 0 protects Segment 0, Ring 1 protects Segment 1, and so on through Ring 7 which protects Segment 7.

The eight segments and their rings are arranged hierarchically. Ring 0 is the most protected domain and is used by the AOS/VS supervisor while ring 7 is the least protected and is generally used by user software. Privileged instructions, such as those that load the Segment Base registers or Purge Address Translation Unit, can only be executed in Ring 0. Execution of privileged instructions in Ring 0 provides a two state (privileged, non-privileged) CPU. Under AOS/VS, Rings 0 through 3 are used by system software while Rings 4 through 7 are used by application/user software. Any attempt by a program to call into rings 0 through 3 without proper validation of the call by the TCB results in a protection violation.

Ring Crossing

A ring cannot be crossed accidentally since the hardware increments only the part of the program counter that does not involve the segment. The program must issue either the subroutine XCALL (eXtended CALL) or LCALL (Long CALL) instruction to attempt an inward ring crossing. The program must issue a wide return (WRTN) instruction to attempt an outward ring crossing. The subroutine call must be inward, toward ring 0, and the return call must be outward, toward ring 7. An outward subroutine call or an inward return call will cause a protection fault. The hardware will only allow a process to transfer to another segment through a set of valid gates.

Gates

Gates provide the means for entry into a particular ring (segment). Each ring has one set of valid gates for inward ring calls. Each set specifies the number of gates, the address of each entry point, and the highest segment that can use each gate (for example, the value 3 in the highest segment field specifies that only processes in ring 3 or lower can access the gate). The highest segment number and the segment the gate is specified in define the ring bracket. When a process issues a ring crossing call (LCALL,XCALL), the hardware checks its ring number against the ring bracket for the gate. If the caller's ring number is less than or equal to the upper bound of the ring bracket and greater than the lower bound of the ring braket, the call is executed successfully; otherwise a protection fault occurs.

Loading Programs into Other Rings

AOS/VS provides user programs (which by default are placed in ring 7) with the capability of loading other programs into other user segments (rings 4-6) when ring 7 is already loaded with the initial user program. Loading over previously loaded programs is prohibited. The ?RINGLD system call permits a process to load a program containing a properly defined gate table into any of user rings 4 through 6. The XCALL/LCALL instructions may then be used to access this program. The ?RINGST system call is used to prevent the ?RINGLD call from loading a program into a specified ring. Any attempt to load a program into a system segment (rings 0-3) will result in a protection violation. This may only occur at system initialization time.

Interrupts

The CPU and the operating system maintain the I/O facilities through a hierarchical interrupt system. Any program can initiate an I/O operation by requesting a data transfer to or from a device. The program transmits the request through I/O system calls. The operating system initializes the device and transfers the data using the interrupt system.

The operating system maintains control of the interrupt system by manipulating an interrupt on flag (ION), interrupt mask, and device flags. The interrupt on flag and interrupt mask reside in the processor. The ION flag enables or disables all interrupt recognition while the interrupt mask enables or disables selective device interrupt recognition.

The device flags reside in the device controller and provide the interrupt communication link between the processor and the device. By manipulating the ION flag and the interrupt mask, the interrupt system can ignore all interrupt requests or selectively service certain interrupt requests.

If the ION flag and interrupt mask enable the processor to recognize the interrupt request, the processor services the interrupt. To service the interrupt, the processor first determines the action to take on the currently executing instruction, then redefines the interrupt mask, and finally services the interrupt request.

Interrupt Mask

A device is associated with one of 16 bits in the interrupt mask. When the bit equals one, the mask blocks an interrupt request to the processor. When the bit equals zero, the processor services an interrupt request from the device. Since the processor can address more than 16 device controllers, it can use a bit in the interrupt mask for one or more devices.

Instruction Interrupt

Most instructions are noninterruptible because they require only a minimum of CPU execution time. For instructions that require more time, the processor (if required) interrupts the executing instruction, updates the accumulators, and services the interrupt. If the instruction must continue where it left off (resumable instruction), the processor also sets the processor status bit 2 (IRES) to one. After servicing the interrupt, the processor either restarts or resumes the interrupted instruction.

I/O Controllers and Peripheral Devices

Input/Output in ECLIPSE MV Systems

The ECLIPSE MV systems provide three forms of I/O: the data channel (DCH), which provides I/O communications for medium speed devices such as printers or tapes; and, for synchronous communication lines, the burst multiplexor channel (BMC), which is a high speed communications pathway that transfers data directly between main memory and high speed peripherals such as disk controllers and programmed I/O for low speed devices such as asynchronous communication lines. The I/O to memory transfers for both the DCH and BMC always bypass the address translator.

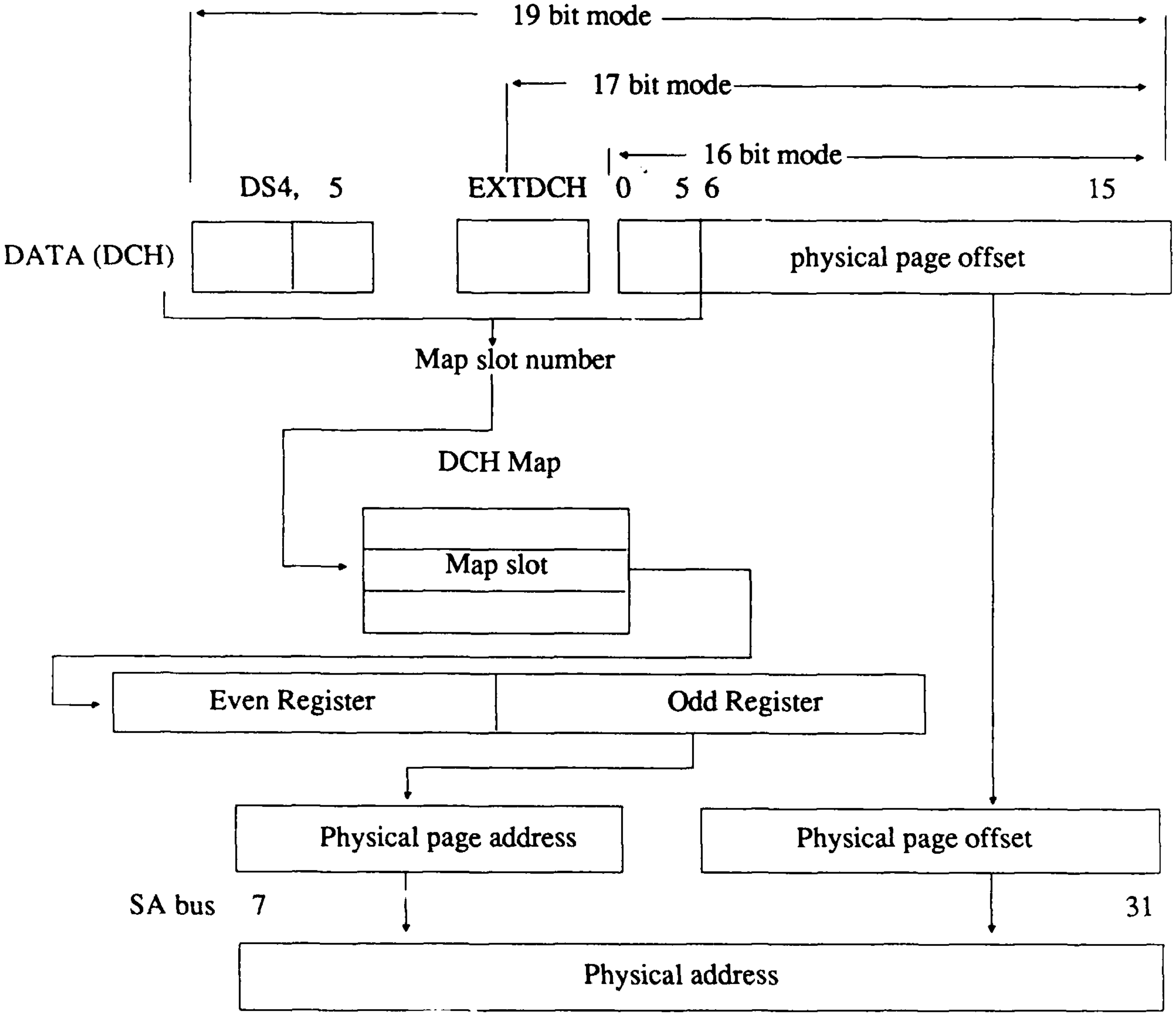
DCH/BMC Maps

The DCH and BMC are each controlled by their respective maps. A map is a series of contiguous map slots containing a pair of map registers: an even-numbered register and a corresponding odd-numbered register (the validity/zero register and the physical page number register). The validity/zero register contains the bits for the translation access. The physical page number register contains the physical page address.

DCH Maps

The data channel map contains 512 map slots. When the data channel transfer specifies a mapped address, six high order data channel address bits and up to three control bits (depending on addressing mode) form a 9-bit address to one of the 512 map slots (see Figure 3). The 15-bit physical page address from this slot combines with the remaining 10 bits of the data channel address to form the 25-bit physical address into main memory. Some devices do not use all three control bits that form the 9-bit slot address. These devices use either no control bits or one control bit forming either a 6-bit or a 7-bit address. These devices are, therefore, restricted to the lower slot numbers corresponding to their addressable ranges.

The CPU uses channel I/O (CIO) instructions for loading or examining the odd or even register of a DCH map slot. The DCH map slots are addressed in the range 4000-5777 (octal). Each pair of even and odd addresses forms one map slot.

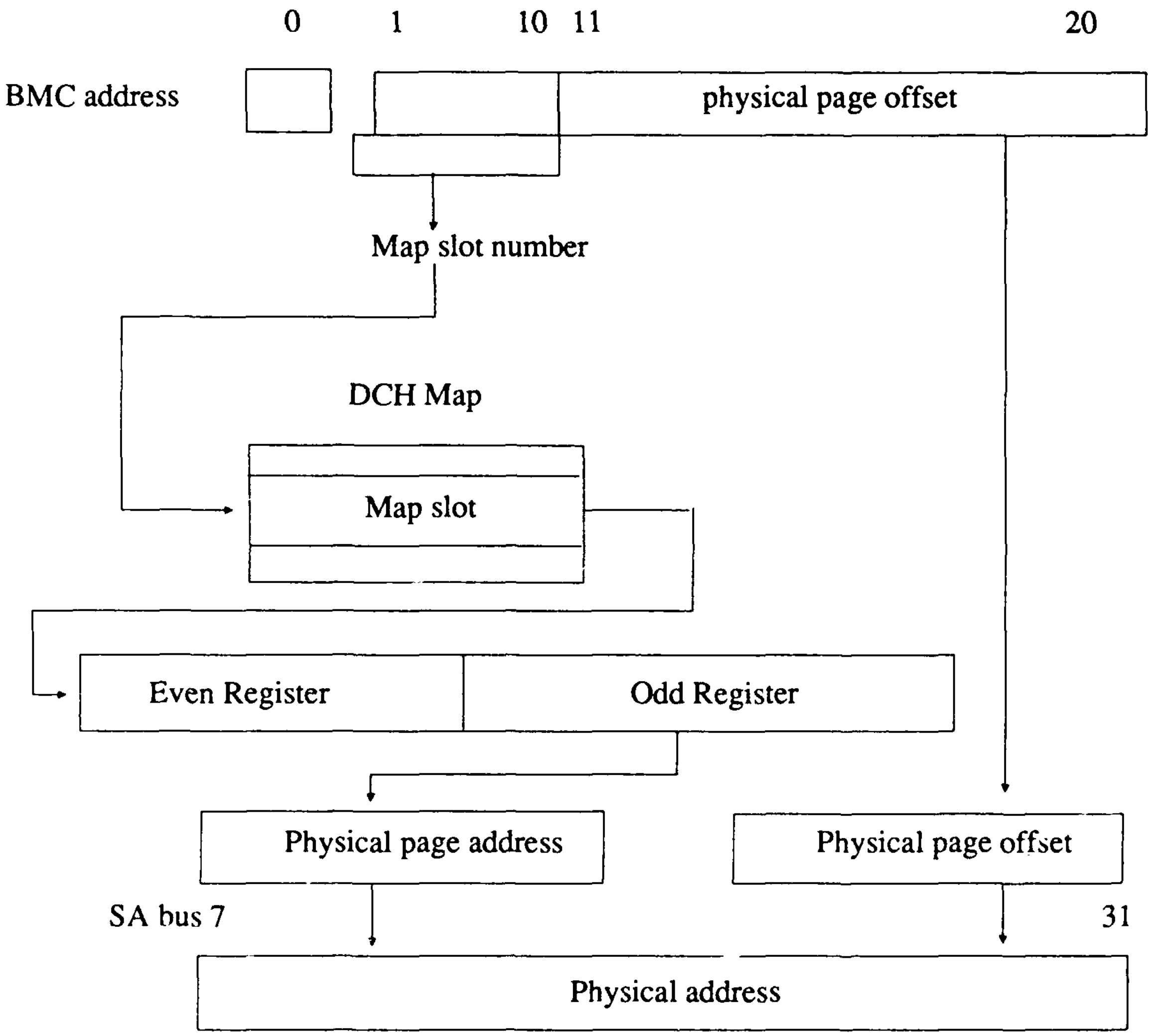


Data Channel Map Address Translation
Figure 3.

BMC Maps

The map table contains 1024 map slots. The BMC map is very similar in organization and function to the data channel map. When the BMC transfer specifies a mapped address, 10 BMC address bits form the address to 1 of the 1024 map slots (see Figure 4). The 15-bit physical page from this slot combines with the remaining 10 bits of the BMC address to form the 25-bit physical address in memory.

The CPU uses the same CIO commands for loading or examining BMC map slots as for DCH map slots. The only difference is in the register address sent with the command. The BMC registers are addressed in the range 000 - 3777 (octal).



BMC Map Address Translation
Figure 4.

Load Effective Address (LEF) Mode

Bit 2 in a Segment Base Register (SBR) determines how the processor will interpret the LEF and I/O operation codes. If bit 2 is set to 1, the processor will execute LEF instructions; otherwise, it will execute I/O instructions. LEF instruction mode is the normal mode of operation. I/O mode allows the user to execute I/O instructions against real devices. Bit 3 in an SBR enables or disables the execution of I/O instructions within that segment. If an I/O instruction is executed in a segment whose SBR has bit 3 set to zero, a protection violation will occur.

User I/O Capabilities

A user may be given the privilege to do physical I/O (i.e., access to user devices) under AOS/VS through the PREDITOR utility (see page 36, "PREDITOR (User Profile Editor)"). This privilege, which is referred to as Access Devices, allows the following:

- Clears the LEF bit in the SBRs for that user enabling the execution of physical I/O instructions as I/O instructions,
- Allows the user to issue an ?IDEF system call

The ?IDEF (interrupt define) system call places the address of a user defined device control table within the system interrupt vector table and defines the DCH/BMC map slots to be used by the device.

A user who has been given the Access Devices privilege also has access to system devices through the physical I/O instructions and can violate system security by reading directly off system devices. In addition, a user with the Access Devices privilege may load microcode into writable control store through the Load Control Store (LCS) instruction. A C2 installation should never grant this privilege to untrusted users.

Device Controller Code

The software which controls the device controllers is considered part of the Trusted Computing Base. Devices are trusted to function correctly and there are no security mechanisms implemented within the devices or controllers. Controllers are not intelligent enough to know about any type of file organization or file access control mechanisms for information on storage devices.

Console Line Controllers

AOS/VS supports several sophisticated and intelligent controllers to handle communications with console devices. These controllers include Input-Output Processors (IOPs) and Intelligent Asynchronous Controllers (IACs).

IOPs are simple processors that allow the CPU to control I/O via the IOC. They are unaccessible by the user as their control signals come directly from the IOC. Remember, the CPU gives the IOC control of the I/O operation. The IOC then communicates with the IOP via the data buses.

An IAC/8 controls up to eight full-duplex asynchronous serial lines and uses the EIA RS-232-C interface standard and modem control logic on each line. An IAC/16 controls up to 16 full-duplex asynchronous, serial lines, using either the EIA RS-232-C or the 20 milli-amp current loop interface standards, but without modem control capability. Each IAC contains a microprogrammed processor with 32 kilobytes of writable local memory, a host interface, and a communications interface. On powerup, an IAC runs a series of confidence tests.

The intelligent code (IACRS) required for the IACs correct operations is downloaded at system initialization time by the Peripheral Manager (PMGR). The primary purpose of these controllers is to minimize the interrupt handling of the primary system CPU. The IACs and IOPs will interrupt the CPU only when a full line of input is available for reading.

The MV/20000 Processor

The MV/20000 processor consists of two identical processors which share all physical memory. One processor is called the mother processor and the other processor is the child processor (master-slave relationship). The two processors run separate processes simultaneously allowing for an increase in performance. Because of the dual processing capabilities, a set of instructions exist that are unique to the MV/20000 to control the processors. This extension of the instruction set is only to control the child processor and not the data passed to and from it.

The caches and Address Translation Units (ATUs) for the two processors are synchronized as follows: each cache monitors the address bus and if data (or addresses in the case of ATUs) is modified, then the modified cache block is invalidated. Only the cache block as identified by the cache tag is invalidated; the rest of the cache remains intact.

Interrupts from a particular device may be deflected to a particular processor through a hardware mechanism in order to achieve interrupt load balancing. A child processor must interrupt the mother processor in order to have any I/O interrupt serviced.

Hardware Peripherals and Components Not Being Evaluated

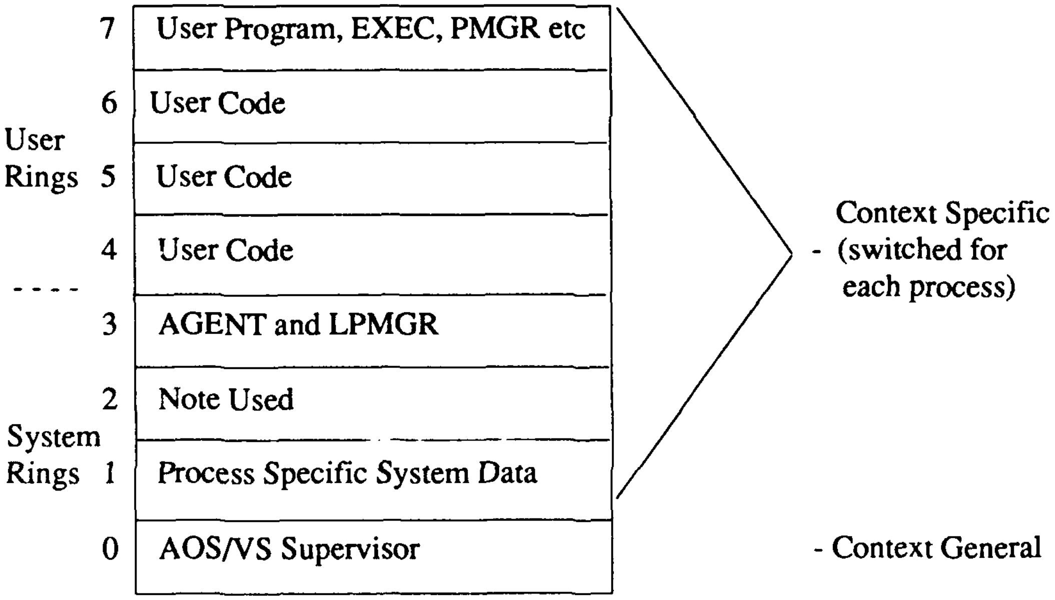
Data General uses the term console to denote user terminals and the term system console to denote the terminal the computer operator uses. The terms console and terminal are interchangeable in this document. The consoles used by an ECLIPSE MV system running AOS/VS are not included as part of the evaluated product nor are any components, controllers and connections used solely for networking. To maintain a C2 rating, consoles and system console connected to the system will not possess local processing capability since they have not been evaluated (i.e., only "dumb" terminals are allowed).

AOS/VS Software Architecture

Data General's AOS/VS is a multiprogramming operating system which runs on the ECLIPSE MV/Family of computers. It is composed of several distinct elements and supports a variety of objects. This section of the report will discuss the TCB in terms of both the supervisor and the trusted processes and the administrative interfaces to the TCB.

AOS/VS TCB

The AOS/VS Trusted Computing Base (TCB) software consists of the following elements: supervisor, AGENT, EXEC, PMGR, XLPT, XPLT, STACKER, OP (Master CLI), and an administrative interface (PREDITOR). Each process context contains the supervisor, AGENT and LPMGR.



	7	User Program, EXEC, PMGR etc	
	6	User Code	
User Rings	5	User Code	
	4	User Code	- Context Specific (switched for each process)
- - - -			
	3	AGENT and LPMGR	
	2	Note Used	
System Rings	1	Process Specific System Data	
	0	AOS/VS Supervisor	- Context General

Segment Contents
Figure 5.

AOS/VS Supervisor

The AOS/VS supervisor consists of portions of AOS/VS which handle the following functions:

- Memory management
- Demand paging
- Process management
- Processor management
- Scheduling
- File and directory management
- ACL handling
- Device drivers
- IPC and connection management
- Interrupt handling

The supervisor is tailored to a specific ECLIPSE MV system and its supporting hardware using the system set-up program, which is called VSGEN. All supervisor code resides in Ring 0 and its services may be requested by, but are not necessarily provided to, any process.

The AGENT

The AGENT represents a code path residing in ring 3 and provides the interface between the user (in rings 4 through 7) and AOS/VS. Every system call made by the user is processed through the AGENT. The AGENT is simply code which resides in ring 3 and makes use of the Validate Word Pointer (VWP) and Validate Byte Pointer (VBP) instructions to verify that byte and word pointers in user packets are valid and not pointers into lower rings. The VBP/VWP instructions compare the ring number in a pointer which is held in one accumulator with the ring specified in another accumulator to determine if the pointer ring is greater than or equal to the other ring number. If not, then the instruction takes an error return. The AGENT will then act accordingly, reflecting the error to the user process. Additionally, the AGENT may preprocess a system call before passing it on to the AOS/VS supervisor or a trusted process for further processing. When the call returns from the

AOS/VS supervisor, EXEC, or PMGR, the AGENT may or may not post-process the call, depending on the service requested. In summary, the AGENT performs the following functions:

- Dispatches all system calls
- Validates user supplied parameter addresses
- Provides extended operating system support
- Deflects system calls to trusted processes (e.g. EXEC and PMGR)
- Provides labeled magnetic tape support
- Provides generic file management support (@LIST, @DATA, @INPUT, @OUTPUT).

System Call Processing

System calls, together with services provided through trusted processes and the system administrative utilities, define the interface which the TCB presents to users of the system. These calls are processed as follows: the user's system call is vectored to the SCALL module, which is bound in with every user program, where it is processed as an LCALL to the AGENT (ring 3). After identifying the call as one that needs the system, the AGENT (ring 3) makes an LCALL to the system (ring 0). The AOS/VS supervisor itself only processes 32-bit calls, so it is the responsibility of the AGENT to convert any 16-bit calls which are issued into 32-bit calls. If a user were to write a personal version of the SCALL module to bypass this checking, it would fail with a protection violation because of the system's use of the gate mechanism (see page 17, "Ring Architecture").

AOS/VS system calls may be classified in three groups: direct calls, where processing of the system call will never pend although the task making the call may pend; parallel calls, where a multi-tasked process may have no other system calls active while this call is active; and expensive calls, which are usually calls to the file system. This implies that AOS/VS may have one or more concurrent paths of execution for any process depending on the system call sequence for a given process. The AGENT maintains a stack for each task in every process, and AOS/VS has a process stack for each ring in which the process resides. All documented system calls are described in *AOS/VS System Call Dictionary* [2]; however, there are a number of calls which are not described in user documentation but are described in documentation which the vendor considers to be proprietary. These calls have been considered by the team during the course of this evaluation and found to fall into three categories: undocumented system calls, which duplicate the functionality of documented calls (these calls have entry points which point to the same code executed in the documented version of the call); undocumented system calls, which may not be used from untrusted code (they contain gates which cause the call to fail if made from any user ring); and "normal" undocumented system calls, which may be executed from user code and perform a unique benign function.

AOS/VS Trusted System Processes

The following processes are defined as system processes and part of the TCB itself:

- Peripheral Manager, PMGR
- Operator Process, OP
- EXEC process
- XLPT, XPLT, and STACKER cooperative processes

The Peripheral Manager (PMGR)

All asynchronous communication devices which are not on the data channel or BMC are controlled by the PMGR (Peripheral Manager). The PMGR can be viewed as existing in three places in an AOS/VS system: a local PMGR in ring 3 (part of the AGENT code for every process), a global PMGR in Ring 3 and 7 as PID 1, and in code which is downloaded into the Intelligent Asynchronous Controllers (IACs) or Input/Output Processors (IOPs) of ECLIPSE MV systems. The PMGR supports the following peripheral devices: IOP, IAC, ALMs, plotters, card readers, OPCON, and consoles.

The PMGR is the initial process (PID 1) when AOS/VS system cold-starts. This provides the advantages of having a well defined interface and providing for easier system maintenance, since PMGR code is not "mixed in" with the supervisor code. The disadvantage is that it is slower, since it is treated by the system scheduler as a user process.

Each IAC or IOP board has a small part of PMGR code (IACRS, LACRS, ALPHARS, CPIRS, and IOPRS) downloaded at system initialization time. The communications paths between the 'local' PMGR and the 'global' PMGR components are implemented using shared file I/O. The code within the IOP and IAC boards all communicate with the global PMGR which communicates with the local PMGR.

The OP(erator) Process

The second process spawned when AOS/VS is cold started is PID 2, called the OP(erator) process, otherwise known as the MASTER CLI. This process comes up on the system console with Superuser, Superprocess, and System Manager privileges and has the username of OP. This is the only process from which EXEC may be initially invoked (unless another process with sufficient ,privileges is PROCed) and has the ability to start or stop auditing under AOS/VS since it has the System Manager privilege, which means it can execute the ?SYSLOG system call.

Any user with the same username as the EXEC process can execute all EXEC commands. This is because the privileges of the OP(erator) are based on the username OP. As an example, many default ACLs for certain directories are set to give the username OP unlimited access. For this reason (among others), the system administrator must be careful not to give the PREDITOR Change Username privilege to any untrusted user.

The operator of an AOS/VS system has physical access to the computer system and all of its components and is a trusted user who performs special functions on behalf of the user community in a time-sharing environment.

The operator is the user who initially brings up the AOS/VS system and runs all software that must be run by a trusted user which includes TBOOT, DFMTR, INSTL, FIXUP, SYSBOOT, and VSGEN. The operator is responsible for creating the first trusted user profile through PREDITOR.

EXEC

AOS/VS EXEC centralizes operator/system manager control over time-sharing, multi-user functions and provides the flexibility and tuning capabilities necessary to manage a multi-user environment. The following are some of EXEC's functions.

- Provides users the capability to log on and off. It checks that the users provide the correct username/password pairs. It then creates a user process using parameters defined in the profile for that user.

- Manages batch streams. EXEC maintains a batch input, list, and output file for each batch request.

- Interprets and spools output to devices like printers.

- Gathers and displays queue status information.

- Handles the magnetic tape MOUNT requests and ACCESS command.

- Fulfills requests from the operator (CONTROL @EXEC) and the user community (?EXEC, ?OPEX).

EXEC resides in Ring 7 and utilizes other AOS/VS programs (processes) such as XLPT, XPLT, and STACKER (the Line Printer, Plotter, and batch job spooling processes, respectively) in order to perform its functions.

The EXEC process (of which there can only be one) can be started only by a process which possesses considerable privileges. The minimum such privileges are Superuser, Superprocess, System Manager, ability to change username, and unlimited sons (see page 33, "Privileges Under AOS/VS"). In the C2 configuration of AOS/VS, the only process to possess all these privileges is PID 2 (the MASTER CLI) when it comes up at the system console on a coldstart.

Most EXEC commands may be issued by a process which has System Manager privileges turned on or any user which has the same username as EXEC (usually OP). A process may also issue EXEC commands to access any devices or queues for which it has discretionary access.

XLPT

XLPT is one of EXEC's cooperative processes which handles queued print jobs. There is one instantiation of XLPT for each printer on the system.

XPLT

XPLT is another of EXEC's cooperative processes; it is responsible for the handling of plotter jobs. There is one XPLT process for each plotter on the system.

STACKER

STACKER is an EXEC cooperative process which is the batch job queuing mechanism. It accepts user input files and interprets them as though they were composed of card images. STACKER then causes EXEC to spawn a process on the behalf of the user who submitted the batch job and submits the card images to the new process as its input deck.

Privileges Under AOS/VS

AOS/VS supports a wide variety of privileges supported by two separate mechanisms. These mechanisms are referred to as "soft" and "hard" privileges. Both hard and soft privileges are defined in the user's profile except for the PMGR privilege which can only be given by EXEC. The hard privileges also reside in the process context data structures.

Soft Privileges

Use console — This privilege allows a user to use a console. Without a console, users only have access to batch facilities.

Use batch — This privilege allows a user to use the batch facilities under A OS/VS.

Use modem — This privilege allows a user to use a modem under AOS/VS. This privilege should not be given to any user since modems are not allowed in a C2 configuration.

Use virtual console — Virtual consoles are sessions attached to a remote host. They are used in networking, which is not part of the evaluated configuration. They should not exist on a C2 system.

Access remote files — This privilege relates to networking, which is not part of the evaluated configuration. As such, this should never be allowed on a secure system.

Change password — Allows users to change their own passwords.

Hard Privileges

Superuser — A process with Superuser privilege has access to all possible privileges by virtue of the fact that it can run PREDITOR and grant itself privileges greater than those it may already possess. Superusers may also circumvent all ACLs.

Superprocess

A process with Superprocess privileges can block or terminate any process in the process hierarchy, not just subordinate processes. This includes the ability of terminating the MASTER CLI which will quickly bring the system down. A Superprocess may also change process type and priority.

System Manager

A user process with System Manager privilege can execute all EXEC commands, enable/disable process class scheduling, initialize and release logical processors, specify a microcode file for use on a particular processor, change the locality of user processes, start and stop auditing, and set the system time.

Access devices

This privilege allows a user process to issue physical I/O instructions to devices including system devices. The user can issue the ?IDEF (interrupt definition) system call and define new devices to the system or define new interrupt service routines for existing devices. The user can also load new micro-code.

Change username

This privilege allows the process to change its username. As a consequence, it grants access to any object whose ACL is non-null; all that the user need do is match the username on the ACL, and AOS/VS grants access, since the ACL mechanism under AOS/V S is based on usernames.

PMGR privilege

This privilege allows a user to create and do I/O to files of type "console."

Unlimited sons

This privilege allows a process to spawn an unlimited number of child processes. It has no security-relevant effects, but it does allow a user to dominate system resource usage.

Create without block

This privilege allows a process to execute simultaneously with any of its child processes. Normally, a father process gets blocked when one of its sons is executing. This allows the father process to get swapped to speed up the system. It is not security-relevant.

Use IPC

This privilege allows the users to use the Interprocess Communications (IPC) facility using ?ISEND and ?IREC.

Change priority — This allows a process to alter its priority in the system scheduling routines. It has no security-relevant effects, but it may allow a process to dominate resource usage.

Change process type — This privilege allows a user to determine his process type. A process may be either resident, pre-emptible, or swappable. This privilege is not security-relevant.

Change program file type — This privilege allows a user to run both 16-bit and 32-bit programs simultaneously. It is not security-relevant.

Change working set — This privilege allows a user to increase his working set size. It is not security-relevant.

The following privileges are security relevant and should not be assigned to any users other than the system administrator or operator in a C2 system: Superuser, Superprocess, System Manager, Access Devices, Change Username, and PMGR. In addition, AOS/VS allows several privileges based on usernames PMGR and OP and on PID 1 and 2. Thus, no user should be given the name PMGR or OP. Table 1 below defines trusted processes privileges.

Table 1, Trusted Process Privileges

Trusted Process Name

Privilege type	PMGR	EXEC	XLPT	XPLT	STACKER	OP
SUPERUSER	YES	YES	YES	YES	YES	YES
SUPERPKOCESS	NO	YES	YES	YES	YES	YES
SYSTEM MANAGER	NO	YES	YES	YES	YES	YES
CHANGE USERNAME	NO	YES	YES	YES	YES	YES
ACCESS DEVICESS	YES	YES	YES	YES	YES	YES
UNLIMITED SONS	NO	YES	YES	YES	YES	YES
CREATE W/O BLOCK	NO	YES	YES	YES	YES	YES
USE IPC	YES	YES	YES	YES	YES	YES
CHANGE PRIORITY	NO	YES	YES	YES	YES	YES
CHANGE PROC TYPE	YES	YES	YES	YES	YES	YES
CHANGE PROG TYPE	NO	YES	YES	YES	YES	YES
CHANGE WORK SET	YES	YES	YES	YES	YES	YES
PMGR	YES	YES	YES	YES	YES	YES

Administrative Interfaces

PREDITOR (User Profile Editor)

PREDITOR can only be invoked by a process with Superuser privilege. It provides a sequence of commands to manage user profiles for system users and is the only source for the granting of privileges for users running under AOS/VS (with the exception of OP, which automatically gets a wide assortment of privileges, including Superuser, Superprocess, Access Devices, and System Manager). PREDITOR allows a system administrator to assign the following:

- Hard and soft privileges
- Initial file for IPC message storage
- Initial program to run upon process creation
- Process priority and disk quota

The change username privilege permits change of the username, but PREDITOR ensures that all usernames are unique. Adding a new user to an AOS/VS system requires executing the PREDITOR CREATE command which builds a User Profile Descriptor (UPD) within the :UPD directory. Both the :UPD directory and UPD files have a null access control list which prevents any non-privileged user from accessing them.

Miscellaneous Administrative Utilities and Functions

The AOS/VS computing system provides support for magnetic tapes and mountable, non-file system disk media for only privileged users through the use of ACLs and the ACCESS command. The default ACL for these devices (OP, WARE) allows access to only the OP username. The UP macro may be used to set the ACLs of these devices to allow other privileged users access at system start-up. Untrusted users should not be given access to magnetic tapes or mountable, non-file system disk media in a C2 configuration.

Privileged users can receive exclusive use of a tape drive by restricting access to the MOUNTQ through the ACCESS command. The ACCESS command instructs EXEC to assign exclusive access to the MOUNTQ to a specified user. Only the specified user and processes with the Superuser privilege would then be able to issue mount requests.

The ACL of a disk or diskette unit remains in force only while there is no initialized LDU in the unit. After a disk has been initialized into the file system, the ACL assigned to the LDU with the Disk Formatter takes effect. An initialized LDU is treated like any directory; its ACL can be changed by an owner of the parent directory. To initialize an LDU, a user needs owner access to the LDU, write access to the directory, and execute access to the unit name(s) in directory :PER.

AOS/VS has several utilities that should be run only from the system console by a trusted user. Some of these utilities execute both in a stand-alone environment and in a stand-among environment.

The actions of the utilities executing in the stand-alone environment are not audited. These stand alone utilities include: TBOOT, the tape boot program; DFMTR, the disk formatter which sets up Logical Disk Units (LDUs) and specifies disk ACLs; FIXUP which fixes the disk structure after system crashes; INSTL, which installs an AOS/VS system on a specific disk; and SYSBOOT, the AOS/VS bootstrap utility which loads AOS/VS from disk in a hardware initial program load (IPL) sequence.

Other utilities which should be used with care are VSGEN, the AOS/VS system generation utility; REPORT, the AOS/VS audit file reduction tool; and those backup restore utilities which are disk block oriented and not file oriented (i.e., they concern themselves only with the ACLs of the disk device, not each file). These backup/restore utilities include PCOPY, the physical disk backup/restore utility; and MSCOPY, the modified sector partial disk backup/restore utility.

TCB Protected Resources

The TCB protected resources under AOS/VS include all of the named objects which consist of files, directories, logical disk units, devices (i.e., terminals, disks, printers, plotters, tape units, and diskettes), pipe files, interprocess communication files, queue files, and AOS/VS processes when acted upon.

Subjects

The only subject supported by AOS/VS is the process. A process is created by the ?PROC system call which creates a process table and places it in the process tree (a hierarchy of all the processes in the system). Each process is uniquely identified through its process identification number (PID). The PID is used to compute the process table address. The process table entry defines the process and includes the PID, username, scheduling information, privilege flags, process status, and memory management information. The process table is the key data structure used by the supervisor to perform resource allocation and access mediations.

Processes under AOS/VS may support multi-tasking (up to 32 tasks per process); may be resident, preemptible, or swappable; have a priority ranging from 1 (highest) to 511 (lowest); and have up to 256 channels for I/O. A process requests system services through system calls via a gate in ring 3. The AGENT, which is part of the TCB, validates the system calls, and either services the call itself or communicates with the supervisor to handle the request.

Process Address Space

Ring 0 is the same for every process context containing the process table for each process. Rings 1-7 are different for each process context. Ring 1 contains additional process specific information such as the channel control blocks (data structures to facilitate file I/O) and process extender table (additional data for process management). Ring 2 is unused, and ring 3 contains AGENT and local PMGR. Rings 4-7 contain user code. For each user segment representing user rings 4-7, user code may be divided into shared and unshared partitions. Partitions are a conceptual entity consisting of an integer number of pages; the supervisor performs memory management on a page basis. The unshared partition starts at the lowest logical address of the segment and grows upward. The shared partition starts at the highest logical address of the segment and grows downward towards the unshared partition.

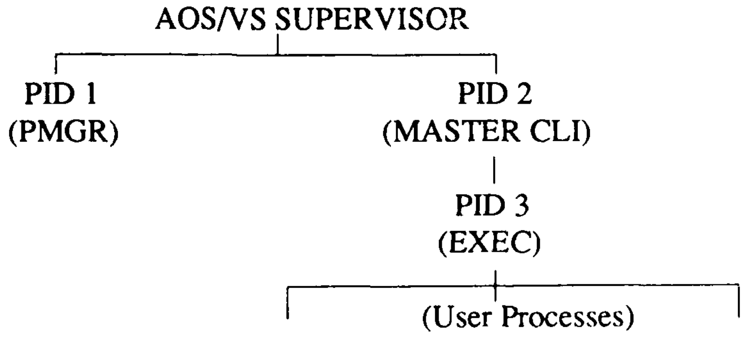
The Process Hierarchy

Figure 6 provides a graphical representation of the process hierarchy. PMGR is always PID 1, and the MASTER CLI (username OP) is always PID 2. EXEC is usually PID 3 with all other processes being subordinate to PID 3.

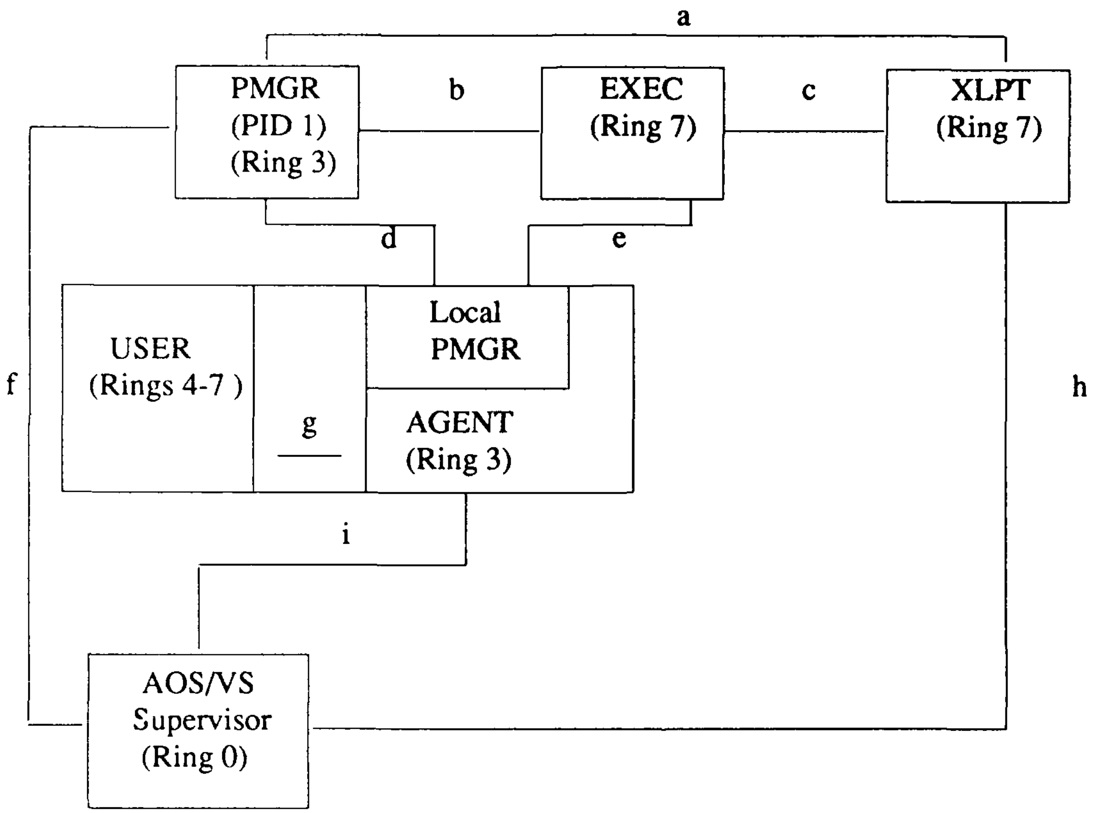
The full process name of a user process consists of the user name and simple process name. The user and simple process names are specified by the process that spawns the user process. It should be noted that a father process that spawns a son process can only specify a user name other than its own if it has the privilege to do so. The default user name is that from the father process. The default simple process name is PROCn, where n is next integer for the series of processes for the user. The operating system uses the user name portion of the full process name to make all the file access decisions.

When a user process is spawned using the ?PROC system call, it inherits a subset of the privileges of the father process. The father process can pass all its privileges to the son process except System Manager privilege.

Figure 7 illustrates the interfaces among the user processes and TCB components. A user process communicates with the TCB through the AGENT (path g). Depending on the user request (system call), the AGENT handles it completely, passes it to the supervisor (path i), passes it to the EXEC (path e), or passes it to the local PMGR. The EXEC handles the request and either queues a request to XLPT/XPLT/STACKER (path c) or outputs data through PMGR (path b). PMGR does the device I/O through the supervisor (path f). Device queue handlers (XLPT, XPLT, STACKER etc.) queue the character I/O through PMGR (path a) whereas they perform block I/O directly through the supervisor (path h).



The Process Hierarchy
Figure 6.



a.	XLPT	⟷	PMGR	-used in printing to non DCH devices
b.	PMGR	⟷	EXEC	-used in logon/logoff
c.	EXEC	⟷	XLPT	-used in queueing print requests, flushes, and restarts
d.	PMGR	⟷	AGENT	-used in handling I/O to terminals
e.	EXEC	⟷	AGENT	-used in the ?EXEC, ?OPEX system calls
f.	PMGR	⟷	AOS/VS	-used for character I/O and reschedule requests
g.	USER	⟷	AGENT	-used in system call processing
h.	XLPT	⟷	AOS/VS	-used in printing to DCH devices
i.	AOS/VS	⟷	AGENT	-used in system call preprocessing

AOS/VS Components - A Conceptual View
Figure 7.

AOS/VS File System

AOS/VS maintains a hierarchical directory and file structure on Logical Disk Units (LDU). LDUs are one to eight physical disk units treated conceptually as one. Directories contain specific information on the various types of user and system files. Files are the information containers for the system and may represent:

- Devices: For each device known to the AOS/VS system there is a corresponding file in the :PER(ipherals) directory representing that device.
- Pipes: A pipe is a special file that implements a FIFO queue accessible to more than one process. A pipe is a transient, byte-oriented file that disappears when all of the processes reading from and writing to the pipe have closed their respective ends.
- IPC files: An IPC file is used for implementing one type of *interprocess communication.*
- Links: A link file is a pointer to another file.
- Queues: The system stores print and batch requests in a queue file.

All file system objects are created with the ?CREATE system call with the type specified in the system call packet. Other types of files exist which are derivations on user files (e.g., a program file), and directories (e.g., control point directory).

System Services

Print and Batch Services

AOS/VS provides the capability of requesting printing services (printer or plotter) or batch processing from a user terminal through CLI commands and from a user program through the ?EXEC and ?OPEX system calls. AOS/VS maintains queues to process print and batch requests.

When a user submits a batch request, CLI creates a temporary file in the user's initial directory which contains the necessary commands to execute the user's request. EXEC then places the request in the batch request queue. Batch requests are processed in an order dependent on the assigned priority and sequence number. When the job comes up for processing, EXEC creates temporary output and list files in the :QUEUE directory. EXEC then spawns a process, which runs with the username and privileges of the requester, to handle the batch request. Upon completion of a job, the temporary files in :QUEUE and the request are deleted.

When a user submits a spool (print) request, a request is added to the request queue in the :QUEUE directory. As in batch, the requests are processed in an order depending on an assigned priority and sequence number. Upon completion of a print job, the request is deleted.

The :QUEUE directory is protected by an ACL which allows users to add files through CLI commands and the ?OPEX and ?EXEC system calls. All requests are processed by EXEC. In the :QUEUE directory, there is a file Q which contains information about all the queues and queue entries. The ACL on the Q file allows users to obtain information about any entries submitted. Users may modify parameters of their own queued entries if the particular queued request is not currently executing. A user may not modify another user's queue entry unless the user's process has the Superuser privilege.

The operator may invoke an AOS/VS utility called QCMP whose purpose is to clean up unused queue entries in the :QUEUE directory. QCMP may not be invoked while EXEC is running.

Interprocess Communication

AOS/VS supports interprocess communication (IPC) to allow processes to communicate with each other. IPC allows processes to transmit variable-length messages from one process to another and sychronize processes during execution.

The message passing IPC method transfers messages between ports. Ports are communication paths that a process specifies by port numbers. There are two types of ports: local and global. Local ports are values that the sender and receiver define to communicate between themselves. Global ports uniquely identify all ports in use in the system and are assigned based upon the process PID, the local port number, and the process's ring number. When a process addresses its local port, AOS/VS translates that port number to the process's global port number.

To send an IPC message, the sender specifies its local port and the global port of the receiver. To receive an IPC message, the process specifies its local port and the global port it wishes to receive from. If a global port is not specified, the process will receive a message from any port. The system calls used are ?ISEND (to send), ?IREC (to receive), and ?IS.R (send and receive). The issuing of the ?ISEND and ?IREC system calls need not be sequential. If a process issues ?IREC and there is no message to receive, the caller will either become suspended or receive an error message (the caller's option). If a process issues an ?ISEND and there is no receiver, the message will either be stored in memory or an error message will be received (also the caller's option). The transfer of an IPC message will only occur if the sender and receiver's port specifications match. This method of IPC requires the Use IPC (?PVIP) privilege to be enabled in the user's profile.

A process must specify a global port number to send and receive IPC traffic from another process. Global port numbers depend on the system environment, and, therefore, frequently change during subsequent process execution. To facilitate the identification of the global port numbers, a second method of IPC uses IPC files. Processes create IPC type files in their default (initial) directory specifying the local port number (with the ?CREATE system call). Senders and receivers may then issue a ?ILKUP system call on the file to determine the global port number.

In order to communicate in this fashion, processes use the IPC file for standard I/O (?READ and ?WRITE) or ?ISEND/?IREC processing. To communicate under standard I/O processing, the owner of the IPC file opens (?OPEN) and issues a read (?READ) on the file. AOS/VS processes the request as a ?IREC system call to the associated port. Only the owner of the file may open the file for input (?READ). If a process who is not the owner opens (?OPEN) the file, a ?ISEND is posted for writing to the IPC file. Since only the owner may open a file for input, it is necessary for two processes to each create and open an IPC file and open the other process's IPC file to establish two way communication. The other method to use the IPC file is to issue ?ISEND and ?IREC system calls directly to and from the IPC file.

The other form of IPC is fast interprocess synchronization. Unlike the message passing IPC mechanisms, no data is moved under fast interprocess sychronization. Fast interprocess synchronization allows tasks within processes to send and receive task-specific signals between the same or different processes. This mechanism requires no privilege and performs no access checks. The system calls used are ?SIGNL (signal another task), ?WTSIG (wait for a signal from another task or process), and ?SIGWT (signal another task and wait for a signal).

Connection Management

Connection management allows a connection to be created between two processes. In a connection, one process is designated the server and the other a customer. In a customer/server relationship, the customer process may use the server process to perform certain actions. Typically, a customer/server relationship occurs when a server is created to perform general services for customer processes. Processes may act as customers and servers simultaneously. Also, a double connection can be created so that two processes are the customer and server of each other.

To become a server, a process must issue a ?SERVE system call (this action is auditable). A process can become a customer by issuing the ?CON system call specifying the server (also auditable). AOS/VS maintains a connection t·ble to manage the exchanges between customers and servers. Each ?CON system call creates an entry in the connection table specifying the PID of the server, PID of the customer, and the customer's ring. The server process determines how to handle each customer. The ?PCNX and ?PRCNX system calls allow a server to pass off the customer to another server without a customer's knowledge.

Connection management allows servers to move bytes to and from a customer's buffers using the ?MBTC and ?MBFC systems calls and use IPC using ?IREC, ?ISEND, and ?IS.R, or the fast interprocess synchronization systems calls ?SIGNL, ?WTSIG, and ?SIGWT. IPC is generally used to notify the customer or server if either has terminated, resigned, or that the connection has been broken. In a customer/server relationship, a privilege is not required to use the first form of IPC.

Consoles

The ownership of a console can be acquired by an explicit assignment (?ASSIGN), an open assignment (?OPEN), or by process assignment (?PROC). When AOS/VS is coldstarted, all consoles are unowned. EXEC will gain ownership of all consoles through the EXEC ENABLE command when it is invoked to bring up the multi-user environment.

Once a process owns the console, it may perform I/O after it issues ?OPENs the console. If it owns the console via a ?PROC, then that console becomes the "process console" and that process is the interruptible owner. Interruptible ownership means that CTRL-C sequences affect the process.

Ownership and interruptible ownership are lost upon termination, reassignment, or passing the console to a son process (not shared). When a console is passed to a son process, the father becomes a pended owner. The father regains ownership when the son process terminates. If an interruptible owner turns on the shared terminal characteristics and passes the console to a son process, the father process remains the interruptible owner.

The owner of a console may grant other processes I/O access to that console. Only a process with I/O access to a console may issue ?READ and ?WRITE. Only the owner of a console may issue control requests such as ?OPEN, ?GCHR/?SCHR (Get/Set terminal characteristics), or ?STOM (Set Device Time Out period). PID 2 has limited control of all consoles.

The only control requests allowed on an unowned console are ?ASSIGN, ?OPEN, or ?PROC.

Any process may issue a ?SEND to any other process console (no access checking). However, a process can turn off ?SEND message receipt via a terminal characteristic. This characteristic cannot suppress messages from PID 2.

The Command Line Interpreter

The Command Line Interpreter (CLI) is the most commonly used interface to AOS/VS. CLI is an interactive programming language, with a macro facility which provides a large repertoire of commands and pseudo-macros to perform numerous functions. Such functions may include the following: using devices such as tapes or printers; create, delete and move files; and execute programs and utilities. The CLI runs as a 16-bit process issuing only 16-bit system calls. CLI runs in each user's address space (in ring 7) as untrusted code and is subject to modification, therefore, each user who invokes the CLI receives a clean copy of the program.

TCB Protection Mechanisms

The following mechanisms are vital to the protection of the AOS/VS TCB and system resources.

Discretionary Access Control

Discretionary access control is an option which is enabled at system generation. AOS/VS uses access control lists (ACLs) to enforce discretionary access control between subjects (processes) accessing objects (i.e., files, directories, logical disk units, devices, pipe files, interprocess communication files, and queue files). ACLs are ordered lists of usernames with their access attributes associated with objects. ACLs may contain up to 255 bytes of access control information; creating a list longer than 255 bytes produces an error. The five access attributes of an ACL which may be given to users are O(WNER), W(RITE), A(PPEND), R(EAD), and E(XECUTE). If none of the attributes is specified for a user, the user has null access, and no access is granted.

Access Attributes

The following are the attributes and their meanings when applied to non-directory files and directory files for a process:

OWNER access to non-directory files allows the following:

- Read and change a file's ACL
- Read the filestatus and permanence attributes of a file
- Set permanence attribute of a file
- Get a complete pathname of a file
- Create a User Data Area (UDA)[1] for the file.

OWNER access to directories allows the following:

- Read and change the ACL of the directory
- Initialize an LDU (that is, format the disk and install a file system on it) if user has OWNER access to root directory
- Rename or delete the directory.

WRITE access to non-directory files allows the following:

- Modify the data in a file
- Read the filestatus and permanence attributes of a file
- Get a complete pathname of a file
- Create a UDA for a file and write to it.

[1] The UDA is a 128-word area associated with a file, but not part of a file, which may contain user data to be acted upon by programs (such as printer control characters).

WRITE access to a directory file allows the following:

- Create, delete, and rename the directory's files
- Read and change the directory's file's ACLs
- Read and change the permanence attribute of files in the directory
- Initialize and release an LDU in the directory.

APPEND access to non-directory files allows the following:

- Read the filestatus and permanence attributes of a file
- Get a complete pathname of a file.

APPEND access to a directory allows the following:

- Add files to the directory
- Initialize an LDU in the directory.

READ access to a non-directory file allows the following:

- Read the data in a file
- Read the filestatus and permanence attributes of a file
- Get a complete pathname of a file
- Read a UDA for a file.

READ access to a directory file allows the following:

- List the name, filestatus, and permanence attribute of all files in the directory
- Use this directory as a working directory
- Read each file's ACL.

EXECUTE access to non-directory files allows the following:

- Execute a file
- Read the filestatus and permanence attribute of a file
- Get a complete pathname of a file.

EXECUTE access to a directory allows the following:

- Use the directory in a pathname
- Make the directory the user's working directory.

To obtain a list of file names in a directory, a user must have EXECUTE and READ access to the directory. A user can delete directory or non-directory files within a parent directory if the user has READ access to the parent directory and OWNER access to the directory's files. One other attribute is the permanence attribute which may be set by those with WRITE access to the file. The permanence attribute prevents users from deleting a non-directory or directory file regardless of the ACL.

By default, users who are not specifically given access have null access. Null access excludes a user from access to any non-directory or directory file. A user may list a directory name with null access to the directory.

Templates for Grouping

Templates may be used in specifying usernames within ACLs and are used as the grouping mechanism. Carefully thought out usernames, along with the wildcard characters, are used to define groups. If a user needs to be taken out of a group, then either that user's username or the group name must be changed. The wildcard characters are as follows:

'*' - match any single character except a period

'-' - match any string without a period (including null)

'+' - match any string (with or without periods) including a null string.

Access Checking

ACLs are stored within the ACL data structure and linked off the file information data structure. The file and ACL data structures may only be manipulated by the system calls for the file system. During an access check, if the ACL data structure exists,[1] the ACL is examined left to right for a match with the username. Upon encountering a match, the access rights for that username are assigned to the process. If the ACL data structure doesn't exist, or no match is found in the ACL, the access rights in the Universal ACL are given to the process. The Universal ACL is the access rights that all usernames have in common. If no Universal ACL is found (no common access rights), then access is denied. ACLs are not sorted in any order; entries are examined in the order that they were added. ACLs cannot be updated and, therefore, must be replaced entirely.

Objects Protected

Files, directories, logical disk units, devices, pipe files, interprocess communication files, and queue files are protected by access control lists. Link files have no access control lists of their own. Access to the file to which the link points is determined by the access control lists on the target file and intermediate directories.

Not all forms of IPC are protected by DAC. File-based IPC may be protected by placing the IPC file in a protected directory. The IPC method using ?ISEND and ?IREC requires that the process possess a privilege, that the two processes be in a customer/server relationship, or that the calls be issued on an IPC file. The final method of IPC by using fast interprocess sychronization does not transfer any data (see page 42, "Interprocess Communication").

Shared memory partitions are pages of physical memory that multiple users can read and possibly modify. There are four ways to map a file into pages of the shared partition. The first way is to explicitly open a file for shared access with the ?SOPEN system call. A second way is through the system calls ?SOPPF (open a protected shared file) and ?PMTPF (permit access to an open, protected shared file), which may also be used to restrict access to the shared file, even to a process with the superuser privilege. The third way to share memory pages is to open a file for shared access with a special form of the ?OPEN system call. After opening the file in the above methods, the ?SPAGE system call is used to map the disk blocks into the shared partition pages. The final way to share memory pages is implicitly through the use of assembler directives to define a shared area. These

[1] If all users have the same access attributes to an object, the ACL data structure is not used, and the common attributes are stored with the file information.

areas become part of the final program file and part of the logical address space of any process that uses that program file. In all four cases, access to the file to be shared is controlled though the ACL.

Object Reuse

AOS/VS ensures that objects allocated to a process do not contain residual data left from a previous process. Object reuse applies only to those objects with a storage capability: physical pages and physical disk blocks. All file system objects are constructed with physical disk blocks, thus ensuring that the file system objects will not contain residual data.

Physical Pages

AOS/VS memory pages cannot be accessed without initialization. Whenever a new page is brought into a process's address space, the AOS/VS supervisor will first zero out a page frame and then do paging if appropriate. If a page fault occurs, the AOS/VS supervisor will zero out the page frame before the disk read is performed.

If a user process issues a ?SSHPT system call to extend the size of a shared partition, the new area will be marked invalid. A protection fault will occur if the new area is accessed before it is mapped via ?SPAGE. The ?SPAGE system call validates the page entry and mark the page as shared. The ?MEMI system call is used to increase (or decrease) the size of the unshared partition. The AOS/VS supervisor zeroes the entire page frame before it becomes part of the process's address space.

Physical Disk Blocks

Whenever a disk is initialized via the ?INIT system call, it becomes part of the file system directory hierarchy. The only way for a non-privileged user to access an initialized disk is through file system elements.

Disk file elements are zeroed upon allocation to a process when creating new elements or expanding the size of existing elements. Acquisition of uncleared disk blocks cannot be accomplished with file positioning calls. A process attempting to position past the end of a file, followed by a read request, will get an end-of-file error and no data will be transferred. A process attempting to position the current file pointer to an empty area of a sparse file, followed by a read request, will receive a buffer of zeroes (no disk access is performed). An attempt to write past the end of file or to an empty area will allocate a new file element which will be zeroed before the user data is written.

Identification and Authentication

To enter the system, users are required to identify themselves by entering a unique[1] username and authenticate their identity by entering a corresponding password. Identification and authentication is not requested of the operator on all ECLIPSE MV systems under evaluation.

All username and password information is placed in user profiles in the :UPD directory which has a null access control list. The passwords may be encrypted through PREDITOR by a process with the superuser or system manager privilege. Only processes with the superuser privilege may directly access the profiles. The EXEC process checks that the username/password pair are correct by comparing the user input with the contents of that user's profile.

Users may change their username or password if provided with the necessary privileges in their user profile. Usernames are made unique, but this username uniqueness may be overridden by processes with the Change Username privilege. The Change Username privilege allows a process to take on another username (and take on any privileges associated with that username).

A user initiating a logon sequence is given 30 seconds to successfully logon by entering a valid username/password pair. If the time limit expires or if the username and password are not a valid pair, a failed logon event is recorded in the audit log and the user is given the "INVALID USERNAME - PASSWORD PAIR" error message. An attempted logon event which failed because of repeated incorrect username/password pairs will lock out the console for 10 seconds or break the connection. The number of attempts (default of 5) allowed before failure, and whether to lock the console, or break the connection, may be set by the system administrator.

An interactive process will never be logged out due to inactivity. AOS/VS does not provide automatic password generation or any password aging capability.

Auditing

SYSLOG is the audit facility within AOS/VS which records user actions and hardware errors. The audit information is recorded in separate audit trail files, SYSLOG and ERROR_LOG, respectively. The SYSLOG and ERROR_LOG pathnames, located in the root directory, may either represent the file or a link to a separate logical device. The SYSLOG file is protected with a null ACL allowing

[1] PREDITOR returns an error message when a system administrator attempts to create a username which already exists. The intention is to maintain unique usernames.

only a process with superuser privilege access. The ERROR_LOG ACL is OP,R allowing the OP process read access.

Audit Record Format

The audit record format includes the record length, a datetime stamp, an audit event code, event code dependent data, an event error code, and a process ID. Each auditable event is represented by an audit event code and, with some of the events, additional information related to the event is recorded (e.g., file name for an open event). All events logged have an error code indicating success or failure of the event. The AOS/VS audit reduction tool, REPORT, correlates the process ID to the username through the process create audit record. The tool also translates event codes into appropriate user friendly text messages (e.g., audit event code 910 is translated to "Process created").

Auditable Events

The audit facility is capable of auditing user actions at two levels of detail. Minimal auditing, the default, audits user account information including console connect time, CPU time, I/O blocks read or written, pages printed, and number of processes created. Full auditing records additional security related information including file access, process creations/terminations, use of privilege and administrative actions, and use of IPC mechanisms. File access events include file open, close, print, rename, create, delete, and ACL changes. Use of privilege and administrative actions include superuser, superprocess and system manager privileges on/off, SYSLOG on/off, and profile modifications. Hardware errors are always recorded.

Auditing Assurances

Auditing should be enabled immediately after PID 2 is initiated. Changing the audit file or stopping the audit function should only be done after terminating all user processes and disabling logons. This allows authorized users to perform file management functions on the SYSLOG file. Interrupting the audit function with user processes on the system could result in a loss of traceability of user actions.

Under minimal auditing, when the LDU containing the audit file fills up, auditing will halt. Under full auditing, the system will continue auditing and will panic if the LDU fills up. AOS/VS provides a CLI macro named CHECK_SPACE.CLI, designed to run every 30 minutes, to measure the disk space left on the LDU on which SYSLOG exists. The macro informs the system console when disk space falls below a specified threshold. The system audit file should not be placed on the same LDU as the SWAP and PAGE directories because audit data is collected rapidly, and the AOS/VS will panic and halt if the SWAP/PAGE LDU fills up.

The auditing mechanism maintains its own set of buffers acquired at system initialization. In a panic situation, these buffers are written to the audit files by the Emergency Shut Down (ESD) code only if the panic code indicates the problem is not in the file system or drivers. If a file system panic occurs, the last set of audit records could be lost.

Audit Trail Management

SYSLOG allows PID 2 (usually the Master CLI), or a process with system manager privilege, to execute the ?SYSLOG system call to start and stop auditing, switch both the system log and error log audit trail files, adjust the detail of logging, and include or exclude soft tape errors (I/O inconsistencies). Processes with the superuser privilege may log events of special interest in SYSLOG by using the ?LOGEV system call.

The REPORT utility provides the capability to examine any audit or error log file including the currently active file. REPORT does not have the capability to examine both SYSLOG and ERROR_LOG simultaneously, and there is no automated technique to merge SYSLOG and ERROR_LOG entries in a chronological order. The REPORT utility provides the ability to selectively review the audit trail based on individual identity and most individual events (e.g., all failed logons, all accesses to a file).

System Initialization

AOS/VS provides a System Manager Interface (SMI) which is simply a program and a set of CLI macros and other aids to help a novice system manager deal with the complexities of bringing up AOS/VS. The SMI program issues ?PROFILE, ?OPEX, and ?EXEC commands.

Part of the system initialization process is to ensure that the system disks are initialized. An LDU whose units are powered up is vulnerable before it is initialized. Anyone at the system console or logged on with username OP can read from it or write to it as a physical device. In order to prevent this, the LDUs which are online must be initialized as soon as possible after starting AOS/VS.

The MV/20000 Processor

AOS/VS is initialized on the mother processor. There is only one copy of AOS/VS in physical memory, and it may be executed simultaneously by both processors.

Processor Synchronization

Processor synchronization for controlled access to global system data bases and resources is accomplished through the use of spin locks (i.e., one processor spins in a tight loop waiting for the release of the resource by the other processor) in the AOS/VS code which accesses that data or resource.

Current Limitations of I/O and Interrupts

Any AOS/VS process which must issue physical I/O instructions or issue the ?IDEF (Interrupt Define) system call must run on the mother processor only, while all other processes run on either the mother or child processor.

All interrupts are handled by the mother processor, although a child processor may interrupt (cross interrupt) the mother processor for required services.

Processor Configuration During System Initialization

At system initialization, AOS/VS attaches the mother processor, which is a job (physical) processor, to a single default logical processor. A logical processor is a scheduling arrangement; it specifies the way in which the job processors attached to it are to schedule processes. The default logical processor specifies standard scheduling. When an attached job processor becomes free, that processor runs the highest priority, ready process.

To identify the physical processors on the system, the hardware returns an ID number called the job processor ID (JPID). The mother processor is 0 while a child processor may have a JPID ranging from 1 to 15.

Similarly, AOS/VS identifies each logical processor on the system by giving it an ID number called an LPID. The default logical processor always has an LPID of 0.

EVALUATION AS A C2 SYSTEM

Discretionary Access Control

Requirement

> The TCB shall define and control access between named users and named objects (e.g., files and programs) in the ADP system. The enforcement mechanism (e.g., self/group/public controls, access control lists) shall allow users to specify and control sharing of those objects by named individuals, or defined groups of individuals, or by both, and shall provide controls to limit propagation of access rights. The discretionary access control mechanism shall, either by explicit user action or by default, provide that objects are protected from unauthorized access. These access controls shall be capable of including or excluding access to the granularity of a single user. Access permission to an object by users not already possessing access permission shall only be assigned by authorized users.

Applicable Features

AOS/VS provides discretionary access control (DAC) between subjects (processes running on behalf of users) and objects (files, directories, logical disk units, devices, pipe files, interprocess communication files, queue files, and AOS/VS processes when acted upon).

Files, directories, logical disk units, devices, pipe files, interprocess communication files, and queue files are protected by access control lists. File-type IPC protected by DAC uses standard peripheral devices controlled by ACLs. For the other types of IPC, DAC is not required. One form uses the ?ISEND, ?IREC, ?IS.R system calls and requires the ?PVIP privilege. Another IPC mechanism is fast interprocess synchronization between processes which requires no privileges and makes no access checks. No data is moved using this method; only signals are sent.

Default ACLs may be specified by the system administrator and are located in the user profile. The default ACL in the profile can be overridden by a user-specified ACL default in the login file or at the start of a login session which would be active for the duration of that session. There is an automatic default ACL of [username,OWARE +,,] which gives the creator all access and everyone else null.

The system defaults for device ACLs are as follows: for disks, diskettes, and tapes, the ACL is [OP,WARE]; for consoles and printers, [PMGR,OWARE]; and for :PER (the peripheral device directory) itself, the ACL is [+,RE]. In a C2 environment, access to tapes and mountable, non-file system disk media is allowed only to privileged users (see page 37, Miscellaneous Administrative Utilities and Functions").

Conclusion

AOS/VS satisfies the C2 Discretionary Access Control requirement.

Additional Requirement (B3)

The following changes are made in this requirement at the B3 level:

CHANGE:

> The enforcement mechanism (e.g., access control lists) shall allow users to specify and control sharing of those objects. These access controls shall be capable of specifying, for each named object, a list of named individuals and a list of groups of named individuals with their respective modes of access to that object.

ADD:

> Furthermore, for each such named object, it shall be possible to specify a list of named individuals and a list of groups of named individuals for which no access to the object is to be given.

Conclusion

AOS/VS satisfies[1] the B3 Discretionary Access Control requirement. AOS/VS access control lists combined with the use of templates for grouping are capable of supporting functional access control to this level of granularity.

[1] Although AOS/VS satisfies this requirement at the B3 level, it does not satisfy the assurance or accountability requirements above its rated level.

Object Reuse

Requirement

All authorizations to the information contained within a storage object shall be revoked prior to initial assignment, allocation, or reallocation to a subject from the TCB's pool of unused storage objects. No information, including encrypted representations of information, produced by a prior subject's actions is to be available to any subject that obtains access to an object that has been released back to the system.

Applicable Features

Object ıeuse applies only to those objects with a storage capability. The AOS/VS operating system supports the following storage objects and ensures that when the objects are allocated they do not contain residual data:

- Tapes and non-file-system disks: These must be cleared by manual procedures. For disks, Data General provides a disk initialization utility, DFMTR, which will write different patterns to every block on the physical disk media in order to locate bad sectors if the user stipulates that "surface analysis" is desired. In writing these patterns, all existing data is destroyed.

- Physical memory pages: Any time a process issues a ?MEMI svstem call to allocate memory to itself, the AOS/VS supervisor zeroes each page before allowing the process to address into them.

- Disk blocks defined to the AOS/VS file system: If a disk has been initialized as part of the AOS/VS file system, it may only be accessed via the AOS/VS directory structure. Disk file elements are zeroed upon allocation and acquisition of uncleared disk blocks cannot be accomplished with file positioning calls.

Conclusion

AOS/VS satisfies the C2 Object Reuse requirement.

Identification and Authentication

Requirement

> The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. Furthermore, the TCB shall use a protected mechanism (e.g., passwords) to authenticate the user's identity. The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user. The TCB shall be able to enforce individual accountability by providing the capability to uniquely identify each individual ADP system user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual.

Applicable Features

Each user is identified to AOS/VS through a unique 1 to 15 character username. The user must also enter a password corresponding to the username to authenticate user identity. Passwords are 6 to 15 characters. Username and password strings are built from the character set "A"-"Z", "0"-"9", "?", "$", "_", and ".". Operators are not required to identify and authenticate themselves to enter the system; therefore, the system console must be physically secured.

By default, EXEC allows 5 logon attempts after which the console will be locked or the connection broken. The system administrator may specify a different number of logon attempts and whether the console should be locked or the line disabled. When a user succeeds in logging on to the system, the user is assigned a PID and all auditable events are associated with that PID and username.

All passwords are stored in the user profile directory, :UPD. The access control list for the :UPD directory is null so that only processes with the superuser privilege may access the user account information. User profiles are created, modified, and deleted via PREDITOR. Passwords may, at the system administrator's discretion, be stored in an encrypted format.

Conclusion

AOS/VS satisfies the C2 Identification and Authentication requirement.

Audit

Requirement

> The TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects. The audit data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data. The TCB shall be able to record the following types of events: use of identification and authentication mechanisms, introduction of objects into a user's address space (e.g., file open, program initiation), deletion of objects, actions taken by computer operators and system administrators and/or system security officers, and other security relevant events. For each recorded event, the audit record shall identify: date and time of the event, user, type of event, and success or failure of the event. For identification/authentication events the origin of request (e.g., terminal ID) shall be included in the audit record. For events that introduce an object into a user's address space and for object deletion events the audit record shall include the name of the object. The ADP system administrator shall be able to selectively audit the actions of any one or more users based on individual identity.

Applicable Features

AOS/VS provides an audit facility capable of recording an audit trail of events including logon/logoff, object accesses, access violations, process creations/terminations, and use of privilege. The audit trail is protected from unauthorized access, modification, and destruction.

The AOS/VS audit facility records all audit information in the :SYSLOG file. Hardware errors are recorded in the :ERROR_LOG file. These pathnames, located in the root directory, may either represent the file or a link to a separate logical device. The access control list for :SYSLOG is null so that only processes with the superuser privilege may access the audit trail. The access control list for :ERROR_LOG is OP,R so that only the OP process may read the error log.

Audit record headers include record length, a date/time stamp, an audit event code, event error code, and owning process ID. Audit records may also contain a variable length body whose size and content depend on the event type. Examples of such information maintained include object identification, process identification, ring number, privilege set, and error code indicators.

The owning process ID within the audit record header can usually be mapped to the owning username. Anytime logging is stopped (when changing the audit file, or starting/stopping the audit facility, etc...), the audit tool loses the capability of being able to associate the process ID with the owning username.

AOS/VS provides REPORT, a report generator, as an audit reduction tool. The REPORT program displays the information found in the SYS_LOG and ERROR_LOG files and provides the system administrator ability to generate various reports based on a number of options.

Conclusion

AOS/VS satisfies the C2 Audit requirement.

System Architecture

Requirement

> The TCB shall maintain a domain for its own execution that protects it from external interference or tampering (e.g., by modification of its code or data structures). Resources controlled by the TCB may be a defined subset of the subjects and objects in the ADP system. The TCB shall isolate the resources to be protected so that they are subject to the access control and auditing requirements.

Applicable Features

The TCB under AOS/VS maintains a domain for its execution which protects it from external interference or tampering through the use of rings, ring 0 privileged instructions, and the virtual memory protection mechanism. The rings have well defined entry points. Rings 0-3 are reserved for the TCB. In addition, several trusted processes reside in user rings 4-7 (see page 17, "Ring Architecture").

Resources controlled by the TCB include all of the subjects and objects in the ADP system (see page 38, "TCB Protected Resources"). Processes are isolated and protected from each other through the use of the virtual memory protection features. All defined objects under AOS/VS are protected by the ACL mechanism with the exception of some forms of interprocess communication (see page 45, "Discretionary Access Control").

Conclusion

AOS/VS satisfies the C2 System Architecture requirement.

Additional Requirement (B1)

The following addition is made in this requirement at the B1 level:

ADD:The TCB shall maintain process isolation through the provisioii of distinct address spaces under its control.

Conclusion

AOS/VS satisfies[1] the B1 system architecture requirement.

[1] Although AOS/VS satisfies this requirement at the B1 level, it does not satisfy the assurance requirements above its rated level.

System Integrity

Requirement

> Hardware and/or software features shall be provided that can be used to periodically validate the correct operation of the on-site hardware and firmware elements of the TCB.

Applicable Features

Data General provides several diagnostic utilities to validate the correct operation of the hardware and firmware elements of the TCB. These utilities include the System Control Processor (SCP), DTOS, MV/SYSTEM/X, and MV/ADEX.

The SCP is a separate microNOVA processor which comes with all ECLIPSE MV systems and provides a useful interface between a user and the ECLIPSE MV system through the system console. SCP Diagnostic Tape Operating System (DTOS) Field Replaceable Units (FRU) tests are available in ROM on all ECLIPSE MV systems except for some MV/4000 models. The SCP DTOS may be used to maintain the ERROR_LOG file which logs all hardware errors which occur (if sufficient disk space exists), runs various CPU and peripheral diagnostics tests, and is used to load and verify CPU microcode. The SCP-DTOS ERRLOG command allows a user to view the ERROR_LOG file to determine the hardware faults that occurred. Each hardware event logged has a date and time stamp associated with it.

The MV/System Exerciser test (MV/SYSTEM/X) is a diagnostic engineering tool to test the capabilities of an ECLIPSE MV system and specified peripherals.

MV Advanced Diagnostic Executive System (MV/ADEX) is a diagnostic tool used on ECLIPSE MV systems. MV/ADEX has the capability of dynamically determining the complete hardware configuration, saving such information in its equipment table, and systematically testing all hardware that Data General can connect to an ECLIPSE MV. MV/ADEX tests all security-relevant system instructions.

Conclusion

AOS/VS satisfies the C2 System Integrity requirement.

Security Testing

Requirement

> The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. Testing shall be done to assure that there are no obvious ways for an unauthorized user to bypass or otherwise defeat the security protection mechanisms of the TCB. Testing shall also include a search for obvious flaws that would allow violation of resource isolation, or that would permit unauthorized access to the audit or authentication data.

Applicable Features

Data General and the NCSC team testing resulted in system acceptance against C2 requirements. The team executed Data General's entire functional test suite and wrote and followed a detailed team test plan to additionally verify the correct operation of the system.

The evaluation team first followed the steps in the vendor test plan. These steps included executing the system integrity tests, generating the system, creating test user profiles, running CONTEST,[1] executing the vendor's batch tests, and executing the vendor's interactive tests.

The batch and interactive tests exercised the security relevant functionality of system calls and CLI commands. These tests focused on the ACL mechanism, privileges, file management, file I/O, memory management, profile creation, auditing, logon, IPC, connection management, process management, and multiple logical processor system calls.

In addition, the evaluation team developed and executed 15 of its own tests to the vendor test suite. These tests included filling up the audit log and observing the results, trying to execute unimplemented machine instructions, observing the default ACL's on system files, and ensuring that disk blocks and memory pages are cleared upon allocation.

Conclusion

AOS/VS satisfies the C2 Security Testing requirement.

[1] CONTEST is an application program used to test the robustness of AOS/VS from revision to revision.

Security Features User's Guide

Requirement

> A single summary, chapter, or manual in user documentation shall describe the protection mechanisms provided by the TCB, guidelines on their use, and how they interact with one another.

Applicable Features

The manual *Learning to Use Your AOS/VS System* [6] is a guide to the novice AOS/VS user who may need instruction on basic features of the AOS/VS system. This manual provides a single summary of the user security features which points to the relevant sections in the manual. This manual is supplemented by th*e AOS/VS System Calls Dictionary* [2], which defines the TCB interface and documents the effects of each system call and its interaction with other system calls.

Conclusion

AOS/VS satisfies the C2 Security Features User's Guide requirement.

Trusted Facility Manual

Requirement

> A manual addressed to the ADP system administrator shall present cautions about functions and privileges that should be controlled when running a secure facility. The procedures for examining and maintaining the audit files as well as the detailed audit record structure for each type of audit event shall be given.

Applicable Features

The manual *How to Generate and Run AOS/VS* [5] in conjurction with the *AOS/VS System Call Dictionary* [2] contains the information necessary to configure and run the AOS/VS system in a trusted fashion. Chapter 16, "System Management Considerations - Security" of *How to Generate and Run AOS/VS* [5], is a stand alone section which describes all of the security features for the installation and maintenance of the system. This chapter does reference other chapters for system generation, auditing, backups, and other applicable security relevant administrator roles. This chapter contains a discussion on the amount of overall security required, C2-level systems, Security Features, User Privileges, Log-on Procedures, Controlling access with ACLs, Auditing, Hardware security features, and protecting the site. In addition to the definition of allowed configuration and management practices, Chapter 16 clearly defines the system security policy and the components not permitted in a C2 system, as well as steps to take for detecting and responding to breaches of security.

Conclusion

AOS/VS satisfies the C2 Trusted Facility Manual requirement.

Test Documentation

Requirement

> The system developer shall provide to the evaluators a document that describes the test plan, test procedures that show how the security mechanisms were tested, and results of the security mechanisms' functional testing.

Applicable Features

Data General's test documentation included a Test Plan, a C2 Functional Test Specification, an Audit Cross Reference, source code for the tests, Expected Function Test Results Summary Tables, and test results. The Test Plan defines subjects and objects, describes the test philosophy, environment, and the mechanisms which are addressed and tested for C2 security issues. The Functional Test Specifications describe documented and undocumented system calls allocated into functional areas and define which system calls are covered by which tests as well as expected test results. The test documentation includes a descriptive preamble which includes the test author, test interfaces, date and revision number, description, notes, errors, and history, as well as the test code. The Audit Cross Reference defines system calls that are audited, the user privilege, programs, events, and event codes. The Expected Function Test Results Summary Table lists the same information as the C2 Functional Test Specification for tests not documented in the C2 Functional Test Specification format. The test results include printouts of the results files for the batch tests and hard copies of the interactive test sessions.

Conclusion

AOS/VS satisfies the C2 Test Documentation requirement.

Design Documentation

Requirement

> Documentation shall be available that provides a description of the manufacturer's philosophy of protection and an explanation of how this philosophy is translated into the TCB. If the TCB is composed of distinct modules, the interfaces between these modules shall be described.

Applicable Features

Data General's philosophy of protection is described in their Security Policy document and in the manual entitled *How to Generate and Run AOS/VS* [5] which includes sections describing subjects, objects, a security features summary, privileges, access control, and hardware security features.

The software TCB protection mechanisms and module interfaces are discussed in the manuals entitled *AOS/VS System Concepts* [3] and *AOS/VS Internals* [1]. The hardware TCB protection mechanisms are described in the manual entitled *Principles of Operations, 32-Bit ECLIPSE Systems* [7].

The Data General AOS/VS TCB interface consists of available system calls and trusted processes. Documented system calls are described in the manual entitled *AOS/VS System Call Dictionary* [2], which includes a summary table of all system calls and a detailed description of each system call. Undocumented system calls are described in internal documentation which is company proprietary.

The manual entitled *AOS/VS System Concepts* [3] includes discussions of virtual memory, processes, files, I/O, IPC, devices, and communications. The internals manual, *AOS/VS Internals* [1] (revision 5.0), includes discussions on the MV architecture, the supervisor, the AGENT, EXEC, PMGR, communications, the CLI, system management, and the user. Data General has provided updates to the team which address the changes made to the system between Revision 5.0 and the evaluated product (revision 7.60).

Conclusion

AOS/VS satisfies the C2 Design Documentation requirement.

EVALUATOR'S COMMENTS

Discretionary Access Control

The AOS/VS discretionary access control system exceeds the requirements in the Criteria by allowing a finer granularity of control. AOS/VS access control lists allow the specification of individual access modes (e.g., read, write) and can control access to objects by named users or groups. However, the group mechanism is restrictive since a group is defined as a partial match on usernames.

Identification and Authentication

Upon successful login, the system outputs a message on the user terminal indicating the time of last login. In addition to protecting authentication data by discretionary access control, passwords may also be encrypted. A mechanism also exists which allows the user only 30 seconds to complete the entry of the login-password pair before resulting in an error message.

Interprocess Communication Protection

In file-based IPC, the file has an access control list associated with it, but it is not enforced. Protection for the IPC file must be provided by placing the IPC file within a protected directory.

Privileges

AOS/VS provides many mechanisms for supplying privileges including: assigned through the user profile (e.g., superuser, superprocess), implicit through the username (i.e., OP, PMGR), and implicit through the Process ID (e.g., PID 1). It would be desirable to have one mechanism for the management of privileges. Many of the trusted processes must execute with all privileges (e.g., Master CLI, EXEC) and, therefore, it is not clear as to what extent least privilege is enforced.

Audit

Even though the system provides individual access modes for discretionary access control, this is not reflected in the audit mechanism. Auditing is performed at file open. If an illegal access mode is attempted after an open, the failure is not audited.

EVALUATED HARDWARE COMPONENTS

The evaluated hardware list includes the ECLIPSE MV of computers (MV/4000, MV/6000, MV/8000, MV/8000 II, MV/8000 C, MV/10000, MV/15000, and MV/20000) and the following peripheral devices:

DISK DEVICES	TAPE DEVICES	PRINTERS	COMMUNICATION BOARDS
6122	6026	4374	4368-A
6061	6125	4373	4367-A
6060	6231	4364	4380
6236	6300	4363	
6237	6299	4327	
6249		4425	
6290		4557	
		4458	
		4322	
		6321	
		4433	
		6215/16	

EVALUATED SOFTWARE COMPONENTS

AOS/VS revision 7.60 is delivered as model 3900. Only model 3900 is being evaluated. The model 3900 product includes ꞌll of the software needed to run AOS/VS in a C2 configuration.

Microcode Files used on ECLIPSE MV Processors

The CPU microcode file required to execute the AOS/VS processor is considered part of the Trusted Computing Base. This includes the latest revisions of microcode files for each processor being considered. The microcode is loaded into writable control store through the Load Control Store (LCS) instruction which is classified as an I/O instruction. The LCS instruction can load up to 16K of control store memory per iteration. Users could load microcode if they have the Access Devices privilege (see page 33, "Privileges Under AOS/VS").

Microcode File Name	Microcode Revision	ECLIPSE MV Processor
MV4000.MCF	10.00	MV/4000
MV4000FP.MCF	10.00	
MV4000G.MCF	10.00	
MV4000GFP.MCF	10.00	
MV6000.MCF	10.00	MV/6000
MV8000.MCF	10.00	MV/8000
MV8000FP.MCF	10.00	
MV8000_II.MCF	10.00	MV/8000 II
MV8000_IIFP.MCF	10.00	
MV8000_C.MCF	8.00	MV/8000 C
MV8000_CFP.MCF	10.00	
MV10000.MCF	5.01	MV/10000
MV10000SX.MCF	5.00	
MV15000.MCF	3.00	MV/15000
MV20000.MCF	9.00	MV/20000

ACRONYMS LIST

ACL	Access Control List
ACx	One of four ECLIPSE MV general purpose registers
AG	Address Generator
ALU	Arithmetic Logic Unit
AOS/VS	Advanced Operating System/Virtual Storage
ATU	Address Translation Unit
BMC	Burst Multiplexor Channel
CIO	Channel I/O
CLI	Command Line Interpreter
CME	Core Memory Entry
CPU	Central Processing Unit
DAC	Discretionary Access Control
DCH	Data Channel
DEC	Decoder
DMA	Direct Memory Access
DoD	Department of Defense
DRP	Diagnostic Remote Processor
DTOS	Diagnostic Tape Operating System
EPL	Evaluated Products List

ESD	Emergency Shutdown Code
FPACx	One of four 64-bit Floating Point Accumulators Registers
FPSR	Floating Point Status Register
FPU	Floating Point Unit
IAC	Intelligent Asynchronous Controller
IOC I/O	Channel Controller
ION	Interrupt On Flag
IOP	Input/Output Processor
IPAR	Initial Product Assessment Report
IPC	Interprocess Communication
IPL	Initial Program Load
JPID	Job Processor ID number
LDU	Logical Disk Unit
LEF	Load Effective Address
LPID	Logical Processor ID number
LPMGR	Local Peripheral Manager
MCU	Memory Control Unit
NCSC	National Computer Security Center
OP	OP(erator) Process
OPCON	Operators Console

PC	Program Counter
PID	Process ID (number)
PIO	Programmed I/O
PMGR	Peripheral Manager
PSR	Processor Status Register
PTE	Page Table Entry
SA	System Address bus
SBR	Segment Base Register
SCP	System Control Processor
SD	System Data bus
SMI	System Manager Interface
TCB	Trusted Computing Base
TFM	Trusted Facility Manual
UDA	User Data Area
UNICORN	Dominant device interface used at Data General
UPD	User Profile Directory
WFP	Wide Frame Pointer register
WSB	Wide Stack Base register
WSL	Wide Stack Limit register
WSP	Wide Stack Pointer register

REFERENCES

[1] *AOS/VS Internals Manual,* Data General Corporation proprietary internal docuument.

[2] *AOS/VS System Call Dictionary,* Data General Corporation, Rev. 3.0, September 1986, 093-000241-03.

[3] *AOS/VS System Concepts*, Data General Corporation, Rev. 01, February 1986, 093-000335-01.

[4] *Introduction to Eclipse MV/20000 Model 1 and 2,* Data General Corporation, Rev. 0, November 1985, 014-001167-00.

[5] *How to Generate and Run AOS/VS*, Data General Corporation, Rev. 7.0, September 1988, 093-000243-07 and addendum 086-000128-00.

[6] *Learning How to Use Your AOS/VS System*, Data General Corporation, November 9, 1988, 069-000031-02.

[7] *Principles of Operation, 32 Bit ECLIPSE Systems,* Data General Corporation, Rev. 5.0, September 1986, 014-000704.

U S GOVERNMENT PRINTING OFFICE 1989 O—940-514

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT UNLIMITED DISTRIBUTION
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) CSC-EPL-89/001	5. MONITORING ORGANIZATION REPORT NUMBER(S) ~~S231,329~~ S 236351
6a NAME OF PERFORMING ORGANIZATION National Computer Security Center 6b. OFFICE SYMBOL *(If applicable)* C12	7a. NAME OF MONITORING ORGANIZATION
6c ADDRESS *(City, State and ZIP Code)* 9800 Savage Road Ft. George G. Meade, MD 20755-6000	7b ADDRESS *(City, State and ZIP Code)*
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS *(City, State and ZIP Code)*	10. SOURCE OF FUNDING NOS

PROGRAM ELEMENT NO.	PROJECT NO	TASK NO.	WORK UNIT NO

11 TITLE *(Include Security Classification)*
Final Evaluation Report Data General Corporation's AOS/VS revision 7.60

12 PERSONAL AUTHOR(S)
Albert C. Hoheb; R. Leonard Brown; Joseph Bulger; Santosh Chokhani; Donald Dasher; Cynthia Grall

13a TYPE OF REPORT Final	13b TIME COVERED FROM TO	14. DATE OF REPORT *(Yr, Mo., Day)* 890222	15. PAGE COUNT 88

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB GR	NCSC DAC

19 ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Data General Corporation Advanced Operating System/Virtual Storage (AOS/VS) revision 7.60 operating system running on the ECLIPSE MV/Family of 32-bit super-minicomputers, has been evaluated by the National Computer Security Center (NCSC). The security features of AOS/VS were examined against the requirements specified by the *DEPARTMENT OF DEFENSE TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA* (Criteria), dated December 1985. The NCSC evaluation team has determined that the highest class at which AOS/VS satisfies all the specified requirements of the Criteria is class C2. A system that has been rated as being a C2 system provides a Trusted Computing Base (TCB) that enforces a discretionary (need-to-know) access control mechanism and audits the security relevant actions of individual users.

This report documents the findings of the evaluation.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL DENNIS E. SIRBAUGH	22b TELEPHONE NUMBER *(Include Area Code)* (301)859-4458	8b OFFICE SYMBOL C12

EDITION OF 1 JAN 73 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE